

Templates

The screenshot shows a dark-themed API documentation page for the `DotnetApiCatalog` class. The left sidebar contains a navigation tree for the `Microsoft.DocAsCode` namespace, including `DataContracts`, `ManagedReference`, and `DotnetApiCatalog`. The main content area has a header "Class DotnetApiCatalog". It includes sections for "IN THIS ARTICLE", "Methods", and "Parameters". The "Methods" section shows a single method: `GenerateManagedReferenceYamlFiles(string, DotnetApiOptions)`. Below it, under "Parameters", is a detailed description of the `configPath` parameter. The "Returns" section indicates a `Task<Unit>` return type with a note about awaiting build completion.

[modern](#) ↗

The modern template

The screenshot shows a light-themed API documentation page for the `ExpandedDependencyMap` class. The left sidebar lists various `Microsoft.DocAsCode` components like `Build`, `Build.J`, and `Comm`. The main content area has a header "Class ExpandedDependencyMap". It includes sections for "IN THIS ARTICLE", "Methods", and "Parameters". The "Methods" section shows a single static constructor: `ConstructFromDependencyGraph(DependencyGraph)`. Below it, under "Parameters", is a detailed description of the `dg` parameter.

[default](#) ↗

The default template

The screenshot shows the DocFX User Manual interface. In the left sidebar, under 'Getting Started', there's a 'Template' section. The main content area shows code snippets for `docfx.json` files. One snippet shows a 'build' section with a 'template' key set to 'custom'. Another snippet shows a 'build' section with a 'template' key set to an array containing 'default' and a path like 'x:/template/custom'. A note below the first snippet says: 'The template path could either be a zip file called <template>.zip or a folder called <template>.' A warning below the second snippet says: 'DocFX has an hardcoded template: default, iframe.html, static.htm 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 for the 'CatLibrary' class. On the left, there's a navigation tree with 'CatLibrary' expanded, showing 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' operator for 'Cat<T, K>'. The declaration is: 'public static int operator -(Cat<T, K> lsr, int rsr)'. Parameters are 'lsr' (Cat<T, K>) and 'rsr' (System.Int32). Returns System.Int32. Below this, there's a section for 'Explicit Interface Implementations' with 'IAnimal.Eat()' and 'IAnimal.Eat(String)' listed.

[mathew](#)

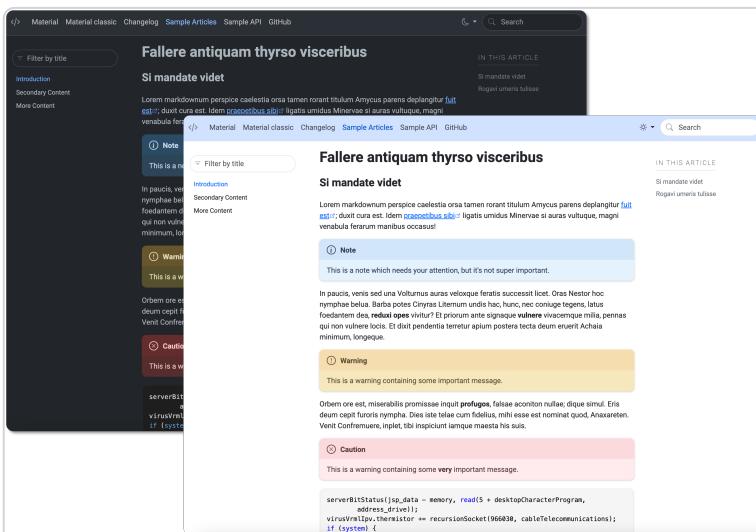
A simple template

docfx.json: "template":

["default", "mathew/src"]

docfx: -t default, mathew/src

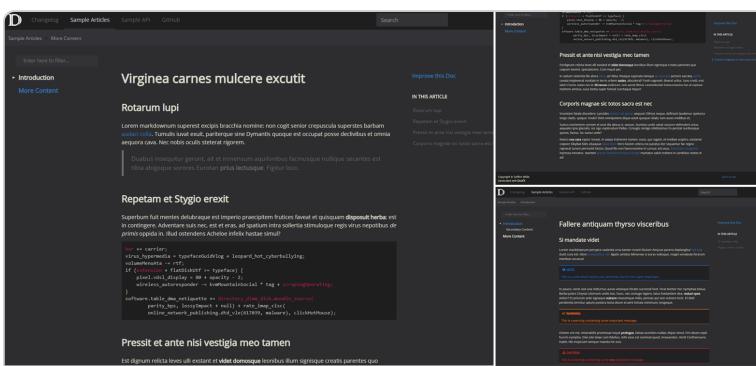
docfx init: git clone
<https://github.com/MathewSachin/docfx-tmpl.git> mathew



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 API. The URL is [Scripting API / Assets.Src / PlayerMovement](#). The page title is "Class PlayerMovement". It describes a basic movement script using WASD/Arcow keyboard input. The "Inheritance" section shows it inherits from L_System.Object and L_PlayerMovement. The "Syntax" section contains the C# code:

```
public class PlayerMovement : MonoBehaviour
```

. The "Fields" section includes a "Speed" field with a description: "Player speed in units per second". The "Declaration" section shows the code:

```
public float Speed
```

. The "FieldValue" section shows the type "System.Single" and the description "Minerva si aurae".

UnityFX

A theme for Unity-esque documentation

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

The screenshot shows a documentation page for the ACME API. The URL is [Articles / Introduction](#). The page title is "Fallere antiquam thyro visceribus". It features a "Si mandate videt" alert box with the message: "Lorem markodorum peripce cœlestia orsa tamen rorant titulum Amycus parens deplanguntur **full est**; duxit cura est. Idem **onaeptibus ubi** ligatis umidus Minervæ si aurae vulnus; magni venabula ferum manibus occasus!" Below it is a note: "This is a note which needs your attention, but it's not super important." A warning box follows: "In paucis, veris sed una **Vollumus** auras veloque feras successit licet. Oras Nestor hoc nymphæ belua. Barba potes Cinyras Literum undis hac, hunc; nec conuge legens, fatus fedelitem des, **reduxi opes vinitur?** Et primum ante signaque **vulnera** vivissemque illata, penitus qui non vulnera locis." A tip box is present: "This is a tip." A warning box: "This is a warning containing some important message." A note: "Orben ore est, miserabilis promissa impul profuga, falsa aconiton nullar; dique simul. Eris deum cepti furoris nymphæ. Dies iste telus cum fidellus, mihi esse est nominat quod; Anareton. Venit Confrersera, inquit, ibi resipicunt tempesta mœsta his aulis." A note: "This is a warning containing some **very** important message." At the bottom, there is a code snippet:

```
# Z_SYNC_FLUSH suffix
ZLIB_SUFFIX = b'\x00\x00\x00\x00\x00\x00\x00\x00'
# Initialize a buffer to store chunks
buffer = BytesArray()
# Create a compression context to run chunks through
inflater = zlib.decompressobj()
```

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 navigation links for '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', 'SetCurrent', and several 'IPath' and 'IAbsolutePath' methods. The main content area has a header 'Method ParseAbsolute'. It contains a 'Declaration' block with code: `public static IAbsoluteDirectoryPath ParseAbsolute(ReadOnlySpan<char>, PathOptions)`. Below it are 'Parameters' and 'Returns' tables. The 'Parameters' table has two rows: 'path' (ReadOnlySpan<char>) and 'options' (PathOptions). The 'Returns' table has one row: 'IAbsoluteDirectoryPath'. There are also sections for 'ParseAbsolute(ReadOnlySpan<Char>, PathFormat, PathOptions)' and 'Declaration' with its own code block.

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](#)

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. The left sidebar lists categories: Pet Store API (pet, store, user) and Contacts API. The main content area has a search bar and a filter dropdown. It shows the Pet resource with a description: "Description for pet tag". Below it is the AddPet operation, which adds a new pet to the store via a POST /pet request. Parameters include a required 'body' parameter with notes: "Pet object that needs to be added to the store". Responses include a 405 status code with the note: "Invalid input". A note at the bottom says: "NOTE: Add pet only when you need to.". Below this is the UpdatePet operation, which updates an existing pet via a PUT /pet request.

[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.

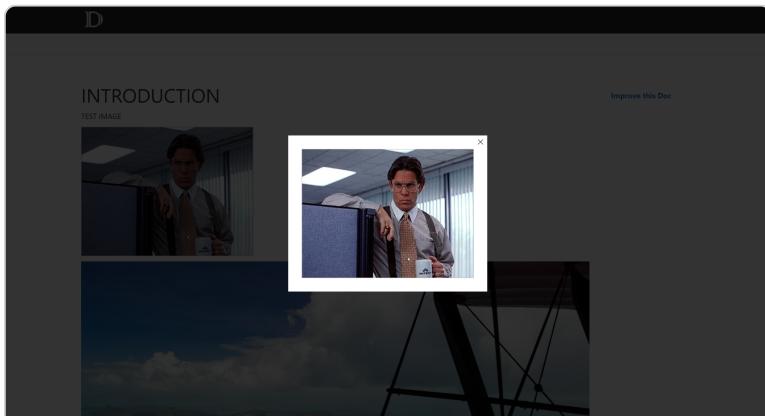
```
docfx.json: template: ["default", "  
<output>/rest.tagpage.<version>/content"]  
docfx: -t default,<output>/rest.tagpage.  
<version>/content  
docfx init: nuget install rest.tagpage -  
OutputDirectory <output>
```

The screenshot shows the Pet Store API documentation for the DeletePet resource. The left sidebar lists categories: Pet Store API (addPet, createUser, deleteUser, findUserByStatus, findPetByTags, getInventory, getOrderById, getPetById, getUserByName, loginUser, logOut, placeOrder, updatePet, updatePetWithForm, updateUser, uploadFile) and Contacts API. The main content area shows the DeletePet resource with a description: "Deletes a pet". Below it is the DeletePet operation, which deletes a pet via a DELETE /pet/{petId} request. Parameters include required 'api_key' (string) and 'petId' (integer). Responses include 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>
```

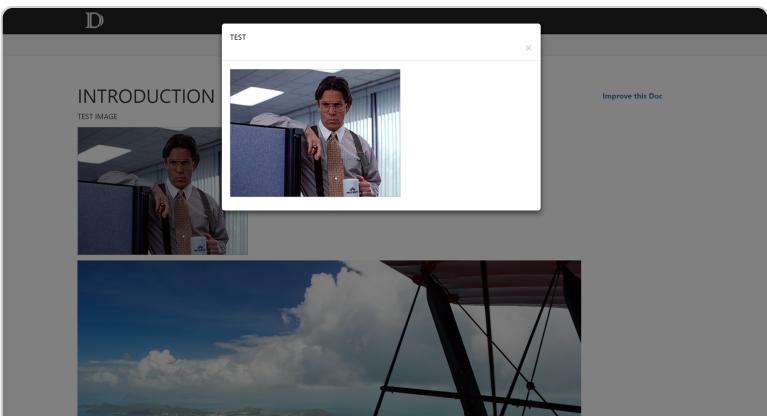


[docfx-lightbox-plugin](#) (Featherlight)

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 interface for a .NET project. The current page is 'Interface ICat'. The left sidebar lists various namespaces and classes, including 'CatLibrary', 'Microsoft.DevDiv', 'MRefDemo.Enumeration', and 'VBTestClass1'. The main content area displays a UML class diagram for 'ICat'. The 'ICat' interface is shown with a red border, and an arrow points from it to the 'Animal' class, which is also shown with a red border. The 'Animal' class has three methods listed: 'Eat', 'EatTool', and 'Excessive'. Below the diagram, there is a section titled 'Inherited Members' with a list of members like 'AnimalName', 'AnimalItem[Int32]', 'Animal.Eat()', etc. At the bottom, it shows the 'Namespace' as 'CatLibrary' and the 'Assembly' as '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
```

```
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:\OMP\userdocs\de\data-size-estimation.md
Translating ...
Saving C:\OMP\userdocs\fr\data-size-estimation.md
Translating C:\OMP\userdocs\de\index.md
Translating ...
Saving C:\OMP\userdocs\fr\index.md
Translating C:\OMP\userdocs\de\plant-production\example.md
Translating ...
Saving C:\OMP\userdocs\fr\plant-production\example.md
Translating C:\OMP\userdocs\de\plant-production\second-file.md
Translating ...
Saving C:\OMP\userdocs\fr\plant-production\second-file.md
Translating C:\OMP\userdocs\de\plant-production\another-dir\another\and-another\something.md
Translating ...
Saving C:\OMP\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 web 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 has a search bar and navigation links: "Documentation", "Getting Started", "Using DocFx and Companion Tools to generate a Documentation website", "Deploy to an Azure Website automatically", "Guidelines", "Working with other decisions", "Working agreements", "Templates", and "Reference". The main content area has a heading "Using DocFx and Companion Tools to generate a Documentation website". Below it is a "Quick Start" section with a note about using a QuickStart folder. A "TIP" section provides steps for quick start using Azure DevOps and Azure App Service. To the right, there's a sidebar with "Improve this Doc" and "IN THIS ARTICLE" sections.

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.