

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. One snippet shows a build configuration with a custom template. Another snippet shows a build configuration with a default template. 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 says: '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

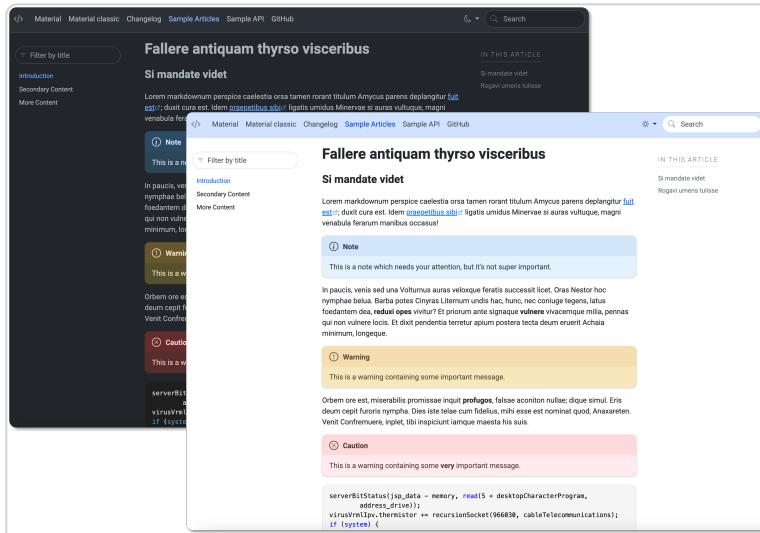
The screenshot shows the API Documentation for the `CatLibrary`. The left sidebar lists classes like `Cat<T, K>`, `CatException<T>`, and `FakeDelegate<T>`. The main content area shows the implementation of the `Subtraction` operator for `Cat<T, K>`. It includes parameters `lsr` and `rsr`, both of type `Cat<T, K>`, and a return type of `System.Int32`. Below this, it shows explicit interface implementations for `IAnimal.Eat()`, which is implemented as `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
```

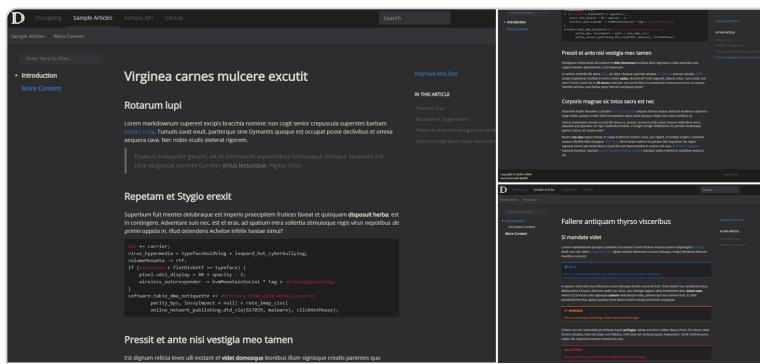


DocFX Material

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 Unity script named `PlayerMovement`. The interface includes a sidebar with navigation links like 'Manual' and 'Scripting API'. The main content area displays the script code:

```

public class PlayerMovement : MonoBehaviour
{
    public float Speed;
}

```

Below the code, there are sections for 'Fields' and 'Field Value'.

UnityFX ↗

A theme for Unity-esque documentation

```

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

```

The screenshot shows a page titled 'Fallere antiquam thyrso visceribus' with a sidebar for 'ACME'. The page contains several note blocks:

- Si mandate videt**: A note with a warning icon.
- This is a note**: A note with a note icon.
- This is a tip**: A note with a tip icon.
- This is a warning containing some important message**: A note with a warning icon.
- This is a note containing some very important message**: A note with a note icon.

At the bottom, there is a snippet of C# code:

```

# 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()

```

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 on a website. The top navigation bar includes links for Home, Articles, API Documentation, and REST API. A search bar is at the top right. The main content area is titled "Pet" under the "Pet Store API". It contains a brief description: "Description for pet tag". There are two sections: "AddPet" and "UpdatePet". The "AddPet" section has a "Request" example: "POST /pet". It shows a table for "Parameters" with one row: "Name": "body", "Type": "string", "Value": "Pet object that needs to be added to the store", and "Notes": "*body". Below this is a table for "Responses" with one row: "Status Code": "405", "Description": "Invalid input", and "Samples": "None". A note says: "NOTE: Add pet only when you need it." The "UpdatePet" section has a "Request" example: "PUT /pet". It shows a table for "Parameters" with one row: "Name": "petId", "Type": "integer", "Value": "Pet id to update", and "Notes": "*petId". Below this is a table for "Responses" with one row: "Status Code": "405", "Description": "Invalid input", and "Samples": "None". On the right side, there are "IN THIS ARTICLE" links: addPet, updatePet, findPetByStatus, findPetByTags, deletePet, getPetById, updatePetWithForm, and uploadFile. There are also "View Source" and "Improve this Doc" buttons.

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.

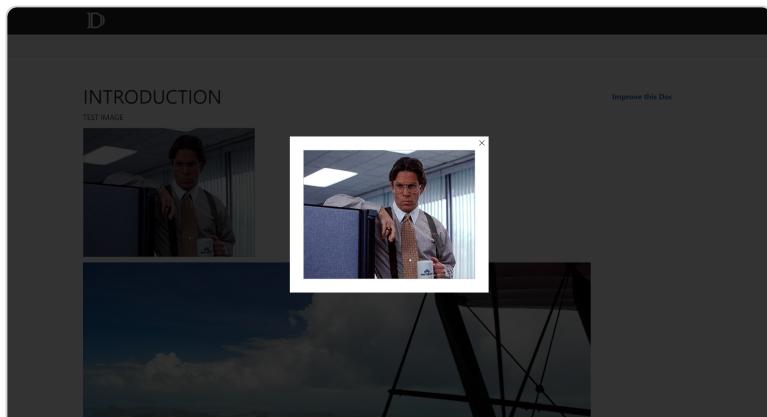
```
docfx.json: template: ["default", "  
<output>/rest.tagpage.<version>/content"]  
docfx: -t default,<output>/rest.tagpage.  
<version>/content  
docfx init: nuget install rest.tagpage -  
OutputDirectory <output>
```

The screenshot shows the Pet Store API documentation on a website, similar to the previous one but with a different URL. The top navigation bar includes links for Home, Articles, API Documentation, and REST API. A search bar is at the top right. The main content area is titled "DeletePet" under the "Pet Store API". It contains a brief description: "Deletes a pet". There are two sections: "DeletePet" and "pet". The "DeletePet" section has a "Request" example: "DELETE /pet/{petId}". It shows a table for "Parameters" with one row: "Name": "petId", "Type": "integer", "Value": "Pet id to delete", and "Notes": "*petId". Below this is a table for "Responses" with two rows: "Status Code": "400", "Description": "Invalid ID supplied", and "Samples": "None"; and "Status Code": "404", "Description": "Pet not found", and "Samples": "None". On the right side, there are "IN THIS ARTICLE" links: pet and >deletePet. There are also "View Source" and "Improve this Doc" buttons.

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>
```

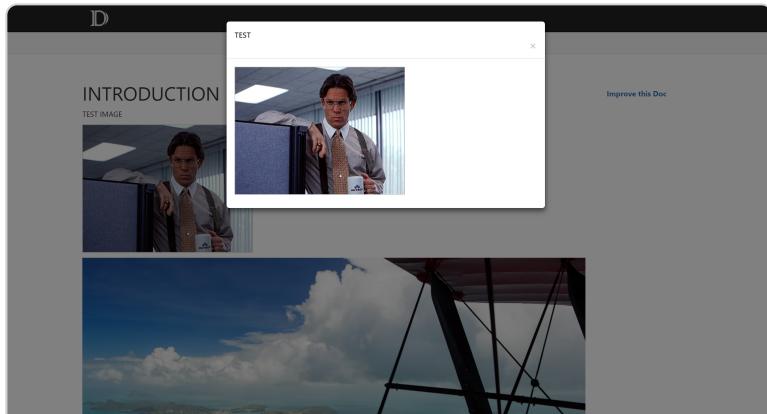


[docfx-lightbox-plugin](#) (Featherlight) ↗

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 'Interface ICat' page in the API Documentation. The interface has a single method named 'Eat'. Below the interface, there is a section for 'Inherited Members' which lists 'Animal.Name' and 'Animal.Eat<String>'. The left sidebar shows a navigation tree with categories like 'CatLibrary', 'Microsoft.DevDiv', 'MRef.Demo.Enumeration', and 'VBTestClass1'.

[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 has sections for "Documentation", "Getting started", "Reference", and "Quick Start". The "Getting started" section is expanded, showing sub-links: "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 "Reference" section is collapsed. The main content area has a heading "Using DocFx and Companion Tools to generate a Documentation website". Below it is a "TIP" section with the following text:

To get you started quickly from scratch, a QuickStart folder is provided which can be copied with some common folder, files and settings mentioned in the steps below. Copy the content from the QuickStart folder to your own repository to get started in minutes.

TIP:

If you want a really quick start using Azure DevOps and Azure App Service without reading the what and how, follow these steps:

1. **Azure DevOps:** If you don't have it yet, create a project in Azure DevOps and create a Service Connection to your Azure environment. Clone the repository.
2. **QuickStart folder:** Copy the contents of the QuickStart folder to the root of your repository, except for the docs folder if you already have one, the .gitignore and the README.md.
3. **Azure:** Create a resource group in your Azure environment where the documentation website.

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.