Table of Contents

Home 2

Articles 3

API Documentation

- .NET API 5
- .NET API (markdown) 6
- .NET API (apipage) 7
- REST API 8

docfx-seed

Description

This is a sample docfx documentation project. It contains .NET source code and markdown files. `docfx.json` is the configuration file for running `docfx`. `docfx` will generate a static website as similar to http://docascode.github.io/docfx-seed.

How to run

Under Windows

- Download and unzip docfx.zip to run `docfx.exe` directly!
- Run `docfx` under current repo! Website will be generated under `_site` folder.
- Run any web hosting tool to host `_site` folder, e.g. `docfx serve _site`.

Cross platform and use `dnx`

As a prerequisite, you will need to install DNVM and DNX. ###Quick Start

- `dnvm upgrade` to get the latest dnvm.

- Add feed https://www.myget.org/F/aspnetrelease/api/v2/ to Nuget.config

  > For Windows, the nuget config file is **%AppData%\NuGet\NuGet.config**.

  > For Linux/OSX, the nuget config file is **~/.config/NuGet/NuGet.config**.

- `dnu commands install docfx` to install `docfx` as a command

- Run `docfx` under current repo! Website will be generated under `_site` folder.

- Run any web hosting tool to host `_site` folder, e.g. `docfx serve _site`.

Further information about `docfx`

`docfx` is a tool to generate documentation towards .NET source code and markdown files. Please refer to docfx to get start. The `docfx` website itself is generated by `docfx`!

Getting Started with `docfx`

Getting Started

This is a seed.

`docfx` is an API documentation generator for .NET, currently support C# and VB. It has the ability to extract triple slash comments out from your source code. What's more, it has syntax to link additional files to API to add additional remarks. `docfx` will scan your source code and your additional conceptual files and generate a complete HTML documentation website for you. `docfx` provides the flexibility for you to customize the website through templates. We currently have several embedded templates, including websites containing pure static html pages and also website managed by AngularJS.

- Click "View Source" for an API to route to the source code in GitHub (your API must be pushed to GitHub)
- `docfx` provide DNX version for cross platform use.
- `docfx` can be used within Visual Studio seamlessly. **NOTE** offical `docfx.msbuild` nuget package is now in pre-release version. You can also build your own with source code and use it locally.
- We support **Docfx Flavored Markdown(DFM)** for writing conceptual files. DFM is **100%** compatible with *Github Flavored Markdown(GFM)* and add several new features including *file inclusion*, *cross reference*, and *yaml header*.

Namespace BuildFromCSharpSourceCode

Classes

CSharp

Namespace BuildFromCSharpSourceCode

Classes

CSharp

Namespace BuildFromCSharpSourceCode

Classes

CSharp

Swagger Petstore

Describe APIs in Pet Store

pet

Description for pet tag

AddPet

Add a new pet to the store

Request

```
POST /pet
```

Parameters

Name	Type	Default	Notes
*body	Pet		Pet object that needs to be added to the store

Responses

Status Code	Type	Description	Samples
405		Invalid input	

> NOTE: Add pet only when you needs.

UpdatePet

Update an existing pet

Request

```
PUT /pet
```

Parameters

Name	Type	Default	Notes
*body	Pet		Pet object that needs to be added to the store

Responses

Status Code	Type	Description	Samples
400		Invalid ID supplied	
404		Pet not found	
405		Validation exception	

FindPetsByStatus

Finds Pets by status

Multiple status values can be provided with comma separated strings

Request

```
GET /pet/findByStatus?status
```

Parameters

Name	Type	Default	Notes
*status			Status values that need to be considered for filter

Responses

Status Code	Type	Description	Samples
200	Pet[]	successful operation	
400		Invalid status value	

FindPetsByTags

Finds Pets by tags

Muliple tags can be provided with comma separated strings. Use tag1, tag2, tag3 for testing.

Request

```
GET /pet/findByTags?tags
```

Parameters

Name	Type	Default	Notes
*tags			Tags to filter by

Responses

Status Code	Type	Description	Samples
200	Pet[]	successful operation	
400		Invalid tag value	

DeletePet

Deletes a pet

Request

```
DELETE /pet/{petId}
```

Parameters

Name	Type	Default	Notes
api_key			
*petId			Pet id to delete

Responses

Status Code	Type	Description	Samples
400		Invalid ID supplied	
404		Pet not found	

GetPetById

Find pet by ID

Returns a single pet

Request

```
GET /pet/{petId}
```

Parameters

Name	Type	Default	Notes
*petId			ID of pet to return

Responses

Status Code	Type	Description	Samples
200	Pet	successful operation	
400		Invalid ID supplied	
404		Pet not found	

UpdatePetWithForm

Updates a pet in the store with form data

Request

```
POST /pet/{petId}
```

Parameters

Name	Type	Default	Notes
*petId			ID of pet that needs to be updated
name			Updated name of the pet
status			Updated status of the pet

Responses

Status Code	Type	Description	Samples
405		Invalid input	

UploadFile

uploads an image

Request

```
POST /pet/{petId}/uploadImage
```

Parameters

Name	Type	Default	Notes
*petId			ID of pet to update
additionalMetadata			Additional data to pass to server
file			file to upload

Responses

Status Code	Type	Description	Samples
200	ApiResponse	successful operation	

store

Access to Petstore orders

Additional description for store tag

AddPet

Add a new pet to the store

Request

```
POST /pet
```

Parameters

Name	Type	Default	Notes
*body	Pet		Pet object that needs to be added to the store

Responses

Status Code	Type	Description	Samples
405		Invalid input	

> NOTE: Add pet only when you needs.

GetInventory

Returns pet inventories by status

Returns a map of status codes to quantities

Request

```
GET /store/inventory
```

Responses

Status Code	Type	Description	Samples
200	object	successful operation	

PlaceOrder

Place an order for a pet

Request

```
POST /store/order
```

Parameters

Name	Type	Default	Notes
*body	Order		order placed for purchasing the pet

Responses

Status Code	Type	Description	Samples
200	Order	successful operation	
400		Invalid Order	

DeleteOrder

Delete purchase order by ID

For valid response try integer IDs with positive integer value. Negative or non-integer values will generate API errors

Request

```
DELETE /store/order/{orderId}
```

Parameters

Name	Type	Default	Notes
*orderId			ID of the order that needs to be deleted

Responses

Status Code	Type	Description	Samples
400		Invalid ID supplied	
404		Order not found	

GetOrderById

Find purchase order by ID

For valid response try integer IDs with value >= 1 and <= 10. Other values will generated exceptions

Request

```
GET /store/order/{orderId}
```

Parameters

Name	Type	Default	Notes
*orderId			ID of pet that needs to be fetched

Responses

Status Code	Type	Description	Samples
200	Order	successful operation	
400		Invalid ID supplied	
404		Order not found	

user

Operations about user

CreateUser

Create user

This can only be done by the logged in user.

Request

```
POST /user
```

Parameters

Name	Type	Default	Notes
*body	User		Created user object

Responses

Status Code	Type	Description	Samples
default		successful operation	

CreateUsersWithArrayInput

Creates list of users with given input array

Request

```
POST /user/createWithArray
```

Parameters

Name	Type	Default	Notes
*body	User[]		List of user object

Responses

Status Code	Type	Description	Samples
default		successful operation	

CreateUsersWithListInput

Creates list of users with given input array

Request

```
POST /user/createWithList
```

Parameters

Name	Type	Default	Notes
*body	User[]		List of user object

Responses

Status Code	Type	Description	Samples
default		successful operation	

LoginUser

Logs user into the system

Request

```
GET /user/login?username&password
```

Parameters

Name	Type	Default	Notes
*username			The user name for login
*password			The password for login in clear text

Responses

Status Code	Type	Description	Samples
200	string	successful operation	
400		Invalid username/password supplied	

LogoutUser

Logs out current logged in user session

Request

```
GET /user/logout
```

Responses

Status Code	Type	Description	Samples
default		successful operation	

DeleteUser

Delete user

This can only be done by the logged in user.

Request

```
DELETE /user/{username}
```

Parameters

Name	Type	Default	Notes
*username			The name that needs to be deleted

Responses

Status Code	**Type**	**Description**	**Samples**
400		Invalid username supplied	
404		User not found	

GetUserByName

Get user by user name

Request

```
GET /user/{username}
```

Parameters

Name	**Type**	**Default**	**Notes**
*username			The name that needs to be fetched. Use user1 for testing.

Responses

Status Code	**Type**	**Description**	**Samples**
200	User	successful operation	
400		Invalid username supplied	
404		User not found	

Other APIs

UpdateUser

Updated user

This can only be done by the logged in user.

Request

```
PUT /user/{username}
```

Parameters

Name	Type	Default	Notes
*username			name that need to be updated
*body	User		Updated user object

Responses

Status Code	Type	Description	Samples
400		Invalid user supplied	
404		User not found	

Definitions

Pet

Name	Type	Notes
category	Category[]	
id	integer (int64)	
name	string	
photoUrls	array	
status	string	pet status in the store
tags	Tag[]	

Category

Name	Type	Notes
id	integer (int64)	
name	string	

Tag

Name	Type	Notes
id	integer (int64)	
name	string	

ApiResponse

Name	Type	Notes
code	integer (int32)	
message	string	
type	string	

Order

Name	Type	Notes
complete	boolean	
id	integer (int64)	
petId	integer (int64)	
quantity	integer (int32)	
shipDate	string (date-time)	
status	string	Order Status

User

Name	Type	Notes
email	string	
firstName	string	
id	integer (int64)	
lastName	string	
password	string	
phone	string	
userStatus	integer (int32)	User Status
username	string	

See Alsos

See other REST APIs:

- Contacts API