**Auteur**

**Photos**

**Titre :** C’est quoi un bon webservces RestFull avec C#

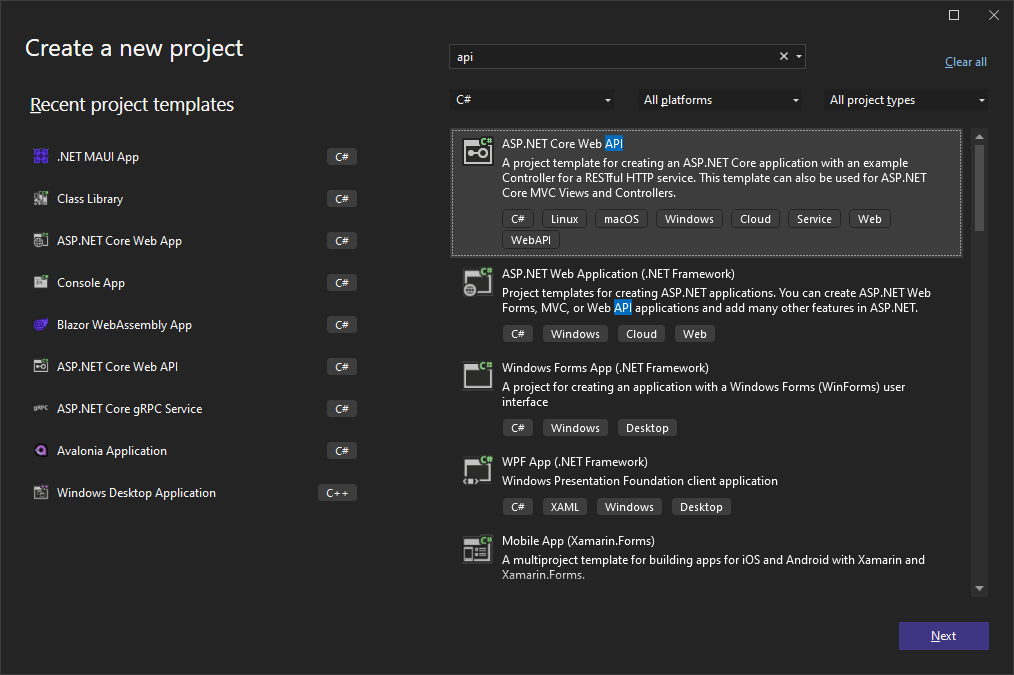
**Sous titre :**

C#, et particulièrement le Framework .NET 6 offre un Framework extrêmement pratique pour développer des webservices RESTFULL conforme aux spécifications OpenAPI. Encore faut-il bien l’utilisé en mettant en œuvre l’ensemble des bonnes pratiques : de documentation, de gestion des erreurs, de logs, de respect des normes.

**Corps Article**

# Création du projet

Avec visual Studio 2022, la création d’un projet de Webservice est très simple. Il faut utilisée le template : « ASP.NET Core API » qui permet d’initialiser Visual Studio avec l’ensemble des composants de base pour un projet de webservices conforme au standard RESTFull HTPP service.



Les options par défaut pour les pages suivantes sont celle dont nous avons besoins (vérifier tout de même que vous etez bien au moins sur du .NET 6 LTS).

Par défaut le package Swashbuckle.AspNetCore est installé pour générer le site d’autodocumentation Swagger. Les Frameworks .NET nécessaire pour le développement de web services sont : Microsoft.AspNetCore.App et Microsoft.NETCore.App.

Si vous lancer l’exécution, vous allez vos retrouver dans votre navigateur avec la documentation Swagger affichée, pour l’exemple de webservice WeatherForecast

Maintenant que le projet est initialisé, nous allons ajouter quelques services ‘oubliés’.

# Premier test

Swagger ui le job est fait pour vous ☺

Mais il faut l’aider à bien le faire.

# Le standard restfull

GET/PUT/POST/DELETE/PATCH

Nommer les fonctions chsarp pour le développeur du ws

Respecte la norme http pour le client du ws

# Créer un contrôleur

Apicontroller

Route

# Implémentation des verbs HTTP

Déclarer les erreurs et valeur de retour

# Déclarer et Respecter le code erreur http

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status200OK)]

# Data annotation Contrôle des données d’entrée

Datat annotation permet de décorer les propertires d’une classe pour assurer le contrôle de données lors des appel rest. Mais aussi dans les interfaces graphiques.

Apicontroller assure le job pour vous si vous utilisé dataannotation

# Logguer le fonctionnement du web service

# Logguer les erreurs de fonctionnement

# Logguer les erreurs lors des appel de webservice

# Versionner son API

Un api ça dure, parfois tres longtemps certe, mais les applications cliente aussi.

Il faut garantir le service dans la durée, et permettre aux applications client de continué a utilisé le service meme si l’API evolue.

# Documenter son API

Mettre de la documentation dans swagger

# Generer la documentation swagger

Documenter via la norme Csharp

Activité la comande dans la solution

Active la generation du xml de documentation

# Générer la documentation help

# Avoir un point de HealthCheck

# References

for information on return value : https://medium.com/awesome-net/web-api-return-types-in-net-94715415ae88

Scott Hanselman. ASP.NET Core RESTful Web API versioning made easy : http://www.hanselman.com/blog/ASPNETCoreRESTfulWebAPIVersioningMadeEasy.aspx

ASP.NET Web API Help Pages using Swagger : https://docs.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger

Data annotations for help : https://learn.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-6.0&tabs=visual-studio// HttpStatusCode Énumération https://docs.microsoft.com/fr-fr/dotnet/api/system.net.httpstatuscode?view=net-6.0

Data annotations attributes : https://learn.microsoft.com/en-us/dotnet/api/system.componentmodel.dataannotations?view=net-6.0

data annotation : https://learn.microsoft.com/en-us/aspnet/web-api/overview/formats-and-model-binding/model-validation-in-aspnet-web-api?source=recommendations

data validation : https://code-maze.com/aspnetcore-modelstate-validation-web-api/

Créer la documentation des Web API ASP.NET Core avec Swagger https://rdonfack.developpez.com/tutoriels/documenter-web-api-aspnet-core-swagger/

Analyseur de valeur de retour : https://learn.microsoft.com/en-us/aspnet/core/web-api/advanced/analyzers?view=aspnetcore-6.0

action asynchrone : https://learn.microsoft.com/en-us/aspnet/core/web-api/action-return-types?view=aspnetcore-6.0

Api controller documentation : https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0#apicontroller-attribute

// Activer le support update partiel d'objet : https://learn.microsoft.com/fr-fr/aspnet/core/web-api/jsonpatch?view=aspnetcore-6.0

// avec le nuget : Microsoft.AspNetCore.JsonPatch

// le jsonpatch est décodée par : t Microsoft.AspNetCore.Mvc.NewtonsoftJson

// article JSON Patch With ASP.NET Core : https://dotnetcoretutorials.com/2017/11/29/json-patch-asp-net-core/