Notes sur le module : *Api en .NET.*

Armel Pitelet

13 janvier 2022

Table des matières

1 Introduction

Définition : **API**.

Application Programming Interface : interface entre un client, un serveur → interface pour l'échange de données vers les apllications.

Définition : **Restfull**.

Representational State Transfert : style d'architechture. En 2000 en parallèle du https 1.1. Framework c# : Asp.Net. Restfull est reconnue fiable et scalable

Remarque : *Les principes REST.*

- Séparation client/serveur : on sépare les responsabilité et on communique via l'API car elle est standardisée. Cela permet de la modularité de chaque coté (on peut faire des modifications de chaque cotés sans que cela impact l'autre.)
- Absence d'état de session (stateless) : l'état d'une session est inclus dans une requête. Les requêtes sont traités de manière isolée. Le serveur n'a pas a mémoriser les informations entre les requêtes (performance, et permet de mélanger de gros volumes de données. L'API ne connait pas les applications avec lesquelles elle discute.)
- Uniformité de l'interface : permet de découpler client et serveur. Permet au serveur de répondre dans un autre format que celui utilisée par l'application (découplage fort pour modularité).
- Mise en cache (caching) : on garde les information régulièrement utilisée en mélmoire (performance). Le cache se fait coté client.
- Architecture en couches (Layered) : Interface uniforme à tt les niveaux. Chaque ressources est identifiée de manière unique et canonicalisée [1] avec son URL. Cela permet d'avoir une API scalable, modulable, et invisible coté client en cas de modifications de l'application.

1. Forme censément la plus simple et en tout cas à laquelle se ramènent toutes les expressions d'un certain type

- Slide n°15 : L'API est ici le point d'entrée à la base de donnée (On a ici un exemple possible d'implémentation : on pourrait avoir EFCore entre l'API et la BDD).

Définition : **URI**.
Uniform Resource Identifier. url + resource (cf shéma slide 16) → la ressource est la requête.

Définition : **Endpoint**.
Point d'entrée de l'API.
- Slide n°17 : Nommage important : le endpoint doit correspondre à une requête métier. Note sur l'exemple le /API est important dans le endpoint.

- Slide n°18 : Les **verbes** http sont les opérations que l'on peut demander à l'API.

- Slide n°20 : Une fois une requête construite et envoyé, c'est le code de retour qui indique si tt c'est bien passé.

- Slide n°21 : Ex 404, le client à crée une mauvaise requête. 5.xx l'erreur provient du serveur.

- Slide n°22 : Le param ou body permet d'envoyer des information complémentaires lors d'une requete. Attention le Endpoint est bien l'entièreté du https :... et pas uniquement le /users/1 (qui ici serait plus la ressource.)

- Slide n°23 : Ici le contenu peut être vide (souvent le cas sur une demande de GET).

- Slide n°24 : Négociation de contenue : ce que la requête envoie (content-type) et ce qu'elle attend en retour (accept). Cette partie est contenue dans le header de la requête.

2 API en ASP.Net

Note :
Controller est le endpoint de l'application. C'est une classe.
Remarque : *Swagger*.
Permet de voir les ressource que l'on a dans l'API. Correspond à la liste des controller.

Note :
Attention les API changent pas mal entre les version de .NET.

Remarque : *Pour créer une API.*
Créer un projet API : ASP with .NET Core. Type d'authentification : pour la sécurité.

Mot clef : **Curl**.
est une interface en ligne de commande, destinée à récupérer le contenu d'une ressource accessible par un réseau informatique.

Remarque : *Pour faire des requêtes directement depuis c#.*
voir : https://docs.microsoft.com/fr-fr/aspnet/core/fundamentals/http-requests?view=aspnetcore-6.0. On utilise l'interface `IHttpClientFactory`.

Remarque : *Pour tester une requete get.*
On peut mettre dans le browser le endpoint et cela effectue automatiquement une requete get.

Remarque : *Pour tester l'API.*
On peut passer par le site **postman** → un peu comme swagger (inclues avec visual studio 2022) mais en plus évolué.

Remarque : *Sur le type de retour de l'API.*
L'API ne retourne pas des classe de base mais des DTO. Les controller doivent renvoyer des DTO.

Remarque : *Sur la durée de vie des controlleurs.*
Lors de la requete l'objet controller est crée, utilisé, puis détruit directement aprés. Il n'y a pas de persistence des controlleurs entre les requetes.

Remarque : *Sur les types de retour dans L'API.*
voir https://docs.microsoft.com/fr-fr/aspnet/core/web-api/action-return-types?view=aspnetcore-6.0, pour la gestion des erreurs et des choses comme sa.

Remarque : *Sur les liaisons de données en ASP.Net.*
POur plus de détail sur [FromBody], [FromQuery] voir https://docs.microsoft.com/fr-fr/aspnet/core/mvc/models/model-binding?view=aspnetcore-6.0.

Remarque : *Sur le découpage d'un projet ASP Net.*
appsetting.json et appsetting.json.development : contient les log de la version release et de la version debug. Dans les properties on a lunchSetting.json qui contient notamment l'url de l'application, le nom du projet, des options diverses, les options swagger.

Remarque : *Dans le Program.cs.*
Les `Service` servent à configurer l'API
Remarque : *Sur les DTO.*
En théorie il faudrait mettre les setter au moins en private (voir en readonly) et gérer via le constructeur. → peut cependnat poser des problèmes avec certaines fonctions de mapping qui viennent se rajouter par dessus.

Note :
Sur les potentiels problèmes de versionning, voir : https://referbruv.com/blog/posts/integrating-aspnet-core-

Remarque : *Sur le mapping.*
On peut utiliser Automappeur pour passer de POCO à DTO facilement. voir https://dev.to/moe23/add-automapper-to-net-6-3fdn.

Remarque : *Pour l'utilisation d'un DbContext.*
voir https://docs.microsoft.com/fr-fr/ef/ef6/fundamentals/configuring/code-based et https://docs.microsoft.com/fr-fr/aspnet/core/data/ef-mvc/intro?view=aspnetcore-6.0.

Remarque : *Dans le DbContext.*
Il faut avoir des property DbContext correspondant aux tables dans la base de donnée.

Acronymes

DSP Nom simple. , *Glossaire :* DSP

Glossaire

DSP la description détaillé.