BARILLOT Estéban

Création et Utilisation des API

Prenons l'exemple d'une TodoList en C#.
https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-8.0&tabs=visual-studio-code

1. Création d'un projet Web

Dans le répertoire du projet

```
dotnet new webapi --use-controllers -o TodoApi
cd TodoApi
dotnet add package Microsoft.EntityFrameworkCore.InMemory
code -r ../TodoApi
```

pour tester le projet

```
dotnet dev-certs https --trust
```

pour démarrer l'application

```
dotnet run --launch-profile https
```

Dans le terminal, s'affichera

```
...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:{port}
...
```

Une fois l'application démarrée cliquez sur le lien "*https://localhost:{port}*"

Dans l'URL "https://localhost:<port>/weatherforecast", dans le fichier JSON le code est similaire à celui-ci :

```
[
      {
      "date": "2019-07-16T19:04:05.7257911-06:00",
      "temperatureC": 52,
      "temperatureF": 125,
      "summary": "Mild"
      },
      {
      "date": "2019-07-17T19:04:05.7258461-06:00",
      "temperatureC": 36,
      "temperatureF": 96,
      "summary": "Warm"
      },
      {
```

```
    "date": "2019-07-18T19:04:05.7258467-06:00",
    "temperatureC": 39,
    "temperatureF": 102,
    "summary": "Cool"
    },
    {
    "date": "2019-07-19T19:04:05.7258471-06:00",
    "temperatureC": 10,
    "temperatureF": 49,
    "summary": "Bracing"
    },
    {
    "date": "2019-07-20T19:04:05.7258474-06:00",
    "temperatureC": -1,
    "temperatureF": 31,
    "summary": "Chilly"
    }
]
```

2. Ajouter un model

Un **modèle** est un ensemble de classes qui représentent les données gérées par l'application.

Pour ajouter un model il suffit de créer le dossier “Model” dans l’application

Créer le fichier “TodoItem.cs” et écrivez le code suivant :

```
namespace TodoApi.Models;

public class TodoItem
{
    public long Id { get; set; }
    public string? Name { get; set; }
    public bool IsComplete { get; set; }
}
```

3. Contexte de base de données

Le **contexte de base de données** est la classe principale qui coordonne les fonctionnalités d'Entity Framework pour un modèle de données.

1. **Ajout**

Créer le fichier suivant “TodoContext.cs” dans le dossier “Model” et entrez le code suivant :

```
using Microsoft.EntityFrameworkCore;
```

```
namespace TodoApi.Models;

public class TodoContext : DbContext
{
      public TodoContext(DbContextOptions<TodoContext> options)
      : base(options)
      {
      }

      public DbSet<TodoItem> TodoItems { get; set; } = null!;
}
```

2. **Enregistrer**

Dans le fichier principal “Program.cs”, mettez à jour votre code à l’aide du code suivant :

```
using Microsoft.EntityFrameworkCore;
using TodoApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();
builder.Services.AddDbContext<TodoContext>(opt =>
      opt.UseInMemoryDatabase("TodoList"));
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
      app.UseSwagger();
      app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

Dans un terminal, pour installer les packages NuGet

```
dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design
dotnet add package Microsoft.EntityFrameworkCore.Design
```

```
dotnet add package Microsoft.EntityFrameworkCore.SqlServer
dotnet add package Microsoft.EntityFrameworkCore.Tools
dotnet tool uninstall -g dotnet-aspnet-codegenerator
dotnet tool install -g dotnet-aspnet-codegenerator
dotnet tool update -g dotnet-aspnet-codegenerator
```

Puis

```
echo 'export PATH=$HOME/.dotnet/tools:$PATH' >> ~/.bashrc
source ~/.bashrc
```

Pour construire le projet

```
dotnet aspnet-codegenerator controller -name TodoItemsController
-async -api -m TodoItem -dc TodoContext -outDir Controllers
```

Mettre à jour la fonction “PostTodoItem”

```
[HttpPost]
public async Task<ActionResult<TodoItem>> PostTodoItem(TodoItem
todoItem)
{
      _context.TodoItems.Add(todoItem);
      await _context.SaveChangesAsync();

      //    return CreatedAtAction("GetTodoItem", new { id =
todoItem.Id }, todoItem);
      return CreatedAtAction(nameof(GetTodoItem), new { id =
todoItem.Id }, todoItem);
}
```

Exécuter l’application, puis dans votre navigateur, dans la fenêtre Swagger, sélectionnez “POST /api/TodoItems” puis “Try it out”. Dans la fenêtre mettez à jour le fichier JSON

```
{
  "name": "walk dog",
  "isComplete": true
}
```

Puis exécuter et taper dans votre URL “https://localhost:<port>/api/TodoItems/1” dans le champ id entrez “1”.