

# Tooling et .NET CLI

## Environnements de développement

Possibilités en matière d'outillage :

- JetBrains Rider (recommandé) : gratuit pour les étudiants et excellent intellisense et autres tooling
- Visual Studio Community (second choix) : gratuit et intellisense moyenne, mais meilleur car c'est un IDE
- Visual Studio Code avec le pack C# : gratuit et léger mais Intellisense très mauvaise + peu de tooling (pas un IDE)

## Projet et Solution

La base d'un projet .NET n'est pas le projet mais la solution. Une solution est une collection de projets. Pour créer une solution :

```
mkdir mon-nouveau-projet
cd mon-nouveau-projet
dotnet new sln
```

La solution est définie par le fichier `.sln` créé par cette commande. On peut ensuite créer des projets. Par convention de structure, on crée au niveau du fichier `.sln` deux dossiers, `src` et `test` dans lesquels seront rangés les projets. Le premier pour le code de production, le second pour le code de test.

Les projets se créent à partir de templates. `dotnet new -l` permet de lister les templates installés. Par exemple pour créer un projet "Application Console" et le mettre dans le dossier `src` par la même occasion : `dotnet new console -o src/MonApp.Console`. On peut ensuite relier le projet à la solution avec `dotnet sln mon-nouveau-projet.sln add src/MonApp.Console/MonApp.Console.csproj`. Le fichier `.csproj` d'un projet est le fichier de configuration du projet au format XML.

Les autres templates de projet qui peuvent être intéressants sont les suivants :

- `classlib` : librairie de classe, pas de point d'entrée
- `xunit` : projet de tests avec le framework de test [XUnit](#)

# Exécution

Les commandes d'exécution sont les suivantes (à partir du répertoire d'un projet) :

- `dotnet run` : exécuter le projet à partir de son point d'entrée
- `dotnet test` : exécuter les tests d'un projet

“ `dotnet test` peut être exécuté depuis le répertoire de la solution pour exécuter tous les tests de la solution

# Librairies

L'outil de gestion de dépendances de .NET (équivalent NPM / Maven / Gradle ...) est [Nuget](#). Pour installer une librairie avec Nuget, la commande `dotnet add package` est disponible. Exemple :

```
dotnet add package Newtonsoft.Json.
```

---

Revision #1

Created 21 November 2024 14:52:47 by Nicolas

Updated 21 November 2024 14:53:18 by Nicolas