Chapitre 3
Les premières étapes avec Angular

1. Introduction

Ce chapitre marque notre entrée dans les fondamentaux du développement Angular, conçu pour vous doter des connaissances et des outils essentiels pour la mise en place d'un environnement de développement structuré. Il faudra exécuter certaines instructions en ligne de commandes mais cela ne requiert pas de connaissances particulières au préalable.

2. Configuration de l'environnement

2.1 Visual Studio Code

Visual Studio Code est un éditeur de code créé par Microsoft et disponible sur Windows, macOS et Linux. Utiliser cet éditeur est un choix personnel pour l'écriture de ce livre. Visual Studio Code est reconnu comme un éditeur efficace, mais aussi le plus utilisé selon une enquête menée par StackOverflow en 2023. Cependant, vous n'aurez aucune gêne à poursuivre la lecture de ce livre en utilisant un autre éditeur de votre choix.

- Pour l'installer, rendez-vous sur la page https://code.visualstudio.com/ afin de télécharger l'installeur. Une fois prêt, nous vous recommandons fortement d'installer au minimum ces trois extensions :
  - **ESLint (*EcmaScript Lint*)** : un outil d'analyse de code statique (ou *lint*), extrêmement populaire, utilisé pour identifier les erreurs, de syntaxe mais aussi de convention de code pour JavaScript/TypeScript. Grâce à lui, en plus de prévenir des erreurs, vous allez pouvoir maintenir la qualité du code et assurer la cohérence de syntaxe/format au sein d'une équipe. L'extension permet à votre éditeur d'analyser votre code en temps réel afin d'alerter et même de corriger automatiquement les erreurs, facilitant ainsi le développement.
  - **Prettier** : un formateur de code, qui prend en charge de nombreux langages. Contrairement à ESLint qui se concentre sur la détection d'erreurs, Prettier est dédié à la mise en forme du code, par exemple l'indentation, les sauts de lignes, les espaces en tabulation ou non, etc. Une fois la configuration décidée, elle sera appliquée à chaque membre de votre équipe ayant installé l'extension. Prettier est aussi capable de corriger les erreurs automatiquement via une commande ou bien à la sauvegarde du fichier, en fonction de votre configuration personnelle.
  - **Angular Language Service** : extension officielle, qui offre une expérience améliorée de développement spécifique à Angular, fournissant des suggestions de code, des complétions automatiques et des diagnostics précis directement dans votre éditeur. Cette extension analyse les applications Angular, comprenant le contexte du code HTML et TypeScript pour fournir des informations pertinentes qui améliorent la productivité et réduisent les erreurs.

Remarque

La mise en place des outils ESLint et Prettier ne se fera qu'au chapitre Compilation et performances. Vous pouvez cependant d'ores et déjà y jeter un œil si vous le souhaitez.

2.2 Node et npm

Node.js est un environnement d'exécution de serveur JavaScript, en dehors des navigateurs. Bien qu'il existe des alternatives comme Deno ou Bun, nous recommandons l'utilisation de Node.js en raison de sa maturité et de sa stabilité éprouvée dans l'industrie. Node, couplé à npm, son gestionnaire de paquets par défaut, forme avec lui un duo puissant pour gérer les dépendances de votre projet et exécuter les tâches essentielles durant les phases de développement.

- Pour l'installer, rendez-vous sur le site officiel https://nodejs.org/en/download et suivez les étapes d'installation.

Remarque

Si vous ne savez pas quelle version installer, choisissez celle qui porte la mention "LTS" (Long Term Support), indiquant que la version est stable et toujours maintenue. Attention, l'installation de npm est incluse. Lorsqu'elle vous sera proposée, veillez à ne pas la désactiver.

- Une fois installé, vous pouvez vérifier que tout s'est bien passé. Pour cela, exécutez la commande suivante dans un terminal de votre choix, un numéro de version devrait vous être affiché en retour :

```
npm -v
```

Il est intéressant de vérifier l'état des nouvelles versions de Node.js sur leur site https://nodejs.org/en/about/previous-releases, pour être conscient des versions dépréciées et savoir à quel moment il est important de mettre à jour votre environnement.

2.3 Introduction à Angular CLI

L'interface de ligne de commande (CLI ou *Command Line Interface*) d'Angular est l'outil incontournable pour les développeurs. En tant qu'assistant de commande polyvalent, elle accélère le processus de développement en automatisant les tâches répétitives, simplifiant ainsi la mise en place, la gestion et l'évolution de projets.

Dans notre cas précis, nous allons commencer par créer un nouveau projet, et bien qu'il soit possible de le construire manuellement, cela peut être fastidieux car, pour un projet complet, nous avons besoin d'un nombre non négligeable d'outils et de fichiers initiaux à créer. L'utilisation de la CLI est préférable pour gagner en fluidité et réduire les erreurs manuelles.

▶ Ouvrez donc un terminal/une invite de commandes, puis installez la CLI via la commande suivante :

```
npm install -g @angular/cli
```

Remarque

La commande `install`, de npm, permet d'ajouter des paquets ; l'argument `-g` signifie "global", indiquant que l'outil est accessible de n'importe où sur votre système, au lieu d'être limité à un projet/dossier précis, étant donné que nous n'en avons pas encore créé.

Maintenant que la CLI est installée, voici quelques commandes basiques à connaître, que nous exécuterons dans les prochaines étapes :

– `ng version` : affiche des informations de version sur notre écosystème.

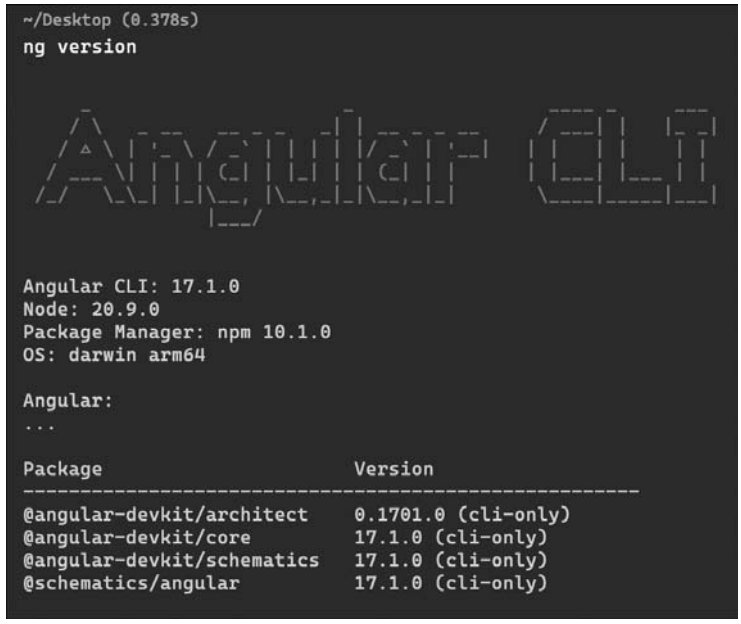
– `ng new` : crée un nouveau projet Angular.

– `ng serve` : compile le projet et lance le serveur local.

▶ Exécutez la première commande pour avoir la confirmation que notre environnement est prêt :

```
ng version
```

La réponse est plutôt complète car elle prend en compte tout l'écosystème nécessaire au bon fonctionnement d'Angular.



Nous pouvons maintenant créer notre première application via la commande suivante, à exécuter dans le dossier où vous souhaitez créer le projet :

```
ng new ma-premiere-app --defaults
```

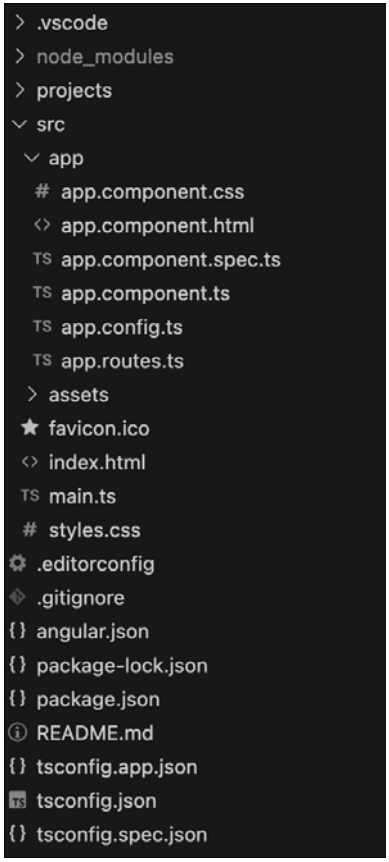
Pour cet exemple, nous utilisons l'argument `--defaults` afin d'utiliser la configuration par défaut. Le projet est maintenant créé et pour voir votre nouvelle application en action, toujours depuis votre terminal, naviguez dans le dossier du projet puis lancez le serveur local avec :

```
ng serve
```

Par défaut, elle rendra votre application accessible via http://localhost:4200.

3. Structure d'un projet Angular

Regardons ensemble ce que la CLI a généré :



Il y a d'une part, à la racine de notre projet, une série de fichiers de configuration qui orchestrent le comportement de l'application, comme les options relatives à npm et TypeScript, certaines réservées aux éditeurs ou à la gestion de version de code. D'autre part, le vif du sujet, le dossier `src` où se trouve le code de votre application.

3.1 Fichiers de configuration

3.1.1 package.json et package-lock.json

Le premier représente la carte d'identité de votre application : vous pouvez y configurer le nom de votre application, sa version, ses packages de dépendances externes, mais aussi des alias de scripts à exécuter. Le second fichier est quant à lui un manifeste généré automatiquement lorsque l'on change les dépendances du projet, et qui documente avec précision les versions de chacun d'eux. Ce fichier est là comme garant lors de l'installation du projet sur différents postes et environnements.

3.1.2 angular.json

Semblable à cela, nous trouverons aussi le fichier `angular.json`. Comme son nom le laisse entendre, il est très spécifique à la configuration d'Angular. Nous pouvons y spécifier des options de compilation, de test, de déploiement, mais aussi la définition de votre espace de travail, car un projet Angular n'est pas forcément composé d'un seul projet. Il peut être une suite d'applications et de librairies. Nous verrons tout cela dans un chapitre dédié.

3.1.3 tsconfig.json

Ensuite, pour la configuration de TypeScript, nous avons `tsconfig.json`, `tsconfig.app.json` et `tsconfig.spec.json`. Ces trois fichiers servent à définir une configuration pour la compilation des fichiers TypeScript. La différence entre ces trois fichiers est très simple. Le premier est la configuration globale dont héritent les deux autres, vous laissant le soin d'y ajouter tout contenu additionnel requis. L'extension `.spec.json` sera récupérée pour la compilation spécifique aux fichiers de tests, tandis que `.app.json` sera pour le code de votre application. L'intérêt est de pouvoir scinder les configurations et de les optimiser au besoin, par exemple, en ayant une configuration moins stricte pour vos tests ou en y incluant/excluant des types externes.

3.1.4 .editorconfig

Ce fichier est automatiquement détecté par certains IDE (*Integrated Development Environment*) (dans le cas contraire, une extension sera requise) et permet d'y uniformiser les configurations à travers plusieurs ordinateurs.

3.1.5 GIT

Pour terminer, il faut savoir qu'une configuration Git sera initialisée dans votre dossier de projet. Git est un système de gestion de versions de fichiers ; il est incontestablement dominant dans l'industrie du développement informatique, ce qui donnera lieu à un dossier caché `.git` et un fichier `.gitignore`. Pour faire court, cet outil sert à créer des versions de votre code lorsque vous le modifiez, ce qui permet de voir l'historique de tous vos changements, ainsi que de partager votre code sur des plateformes comme GitHub ou GitLab avec les différents développeurs de votre équipe. Si vous n'avez pas l'utilité d'un tel outil pour le moment, ne vous n'inquiétez pas, cela n'affectera en rien votre projet.

3.2 Fichiers source

Intéressons-nous au dossier `src`, qui est le cœur de notre application, et où nous passerons la majorité de notre temps de développement. Nous y créerons les différentes composantes d'une application, comme des composants, des services, mais aussi nos ressources autres que le code, comme des médias, des polices d'écriture, etc. Voici donc le contenu du dossier.

3.2.1 index.html et main.ts

Le fichier `index.html` n'est rien de plus qu'une (et la seule) page HTML (*HyperText Markup Language*) statique. Elle sert de support pour amorcer l'application via le second fichier : `main.ts`. Cette amorce est la première chose qu'Angular va faire pour démarrer l'application, l'élément HTML "app-root" va être identifié puis le framework va l'associer au composant `AppComponent`.

3.2.2 AppComponent

Lorsque nous ouvrons le dossier `app`, nous pouvons observer qu'il y a un standard de nommage de fichiers. Dans Angular, les fichiers sont nommés sous la forme "mon-element.type.extension", donc par exemple pour un composant représentant un header, nous allons appeler le template HTML "header.component.html".

En ouvrant le fichier `app.component.ts`, nous pouvons voir qu'il contient une classe TypeScript `AppComponent` avec une instruction `@Component`. Cette instruction est appelée "décorateur" et il est intéressant de noter qu'elle prend en paramètre un objet de configuration qui définit des propriétés telles que `selector`, `templateUrl` et `styleUrls`. Nous examinerons les décorateurs et leur fonctionnement en détail dans un chapitre dédié à TypeScript. À l'intérieur de la classe `AppComponent`, nous pouvons voir une propriété `title`. Notez que dans cette classe, tous les éléments déclarés publics (selon les normes de programmation orientée objet) seront accessibles dans le template HTML.

3.3 Exercice pratique

Passons maintenant à quelque chose de plus interactif. Nous allons modifier notre composant pour y intégrer un compteur simple, un bouton qui incrémente une valeur lors du clic.

- Pour commencer, ouvrez le projet dans votre éditeur de code, puis ouvrez le fichier de template `app.component.html`, supprimez tout le contenu et remplacez-le par un simple bouton. Ce bouton sera l'élément avec lequel l'utilisateur pourra interagir. Ensuite, retournez dans le fichier `app.component.ts`, où nous allons définir la logique derrière le bouton.
- Dans la classe `AppComponent`, ajoutez une propriété `count` initialisée à 0. Cette propriété tiendra le compte des clics sur notre bouton. Ajoutez également une méthode `increment` qui incrémentera la valeur une fois appelée.