

Générer un projet Angular avec Angular-CLI

Démarrer un projet avec Angular-CLI (partie 1)

Par Simon DIENY 

Date de publication : 31 octobre 2018

Dans la première partie de ce tutoriel, nous verrons comment démarrer un projet professionnel avec Angular-CLI. Nous nous consacrerons à la génération du socle avec Angular-CLI, avant d'ajouter dans le prochain tutoriel les éléments indispensables à toute application d'envergure. **Commentez**

En complément sur Developpez.com

- Premier pas avec Angular : un « Hello, World ! » Angular
- Un système d'autocomplétion avec Angular - Afficher des suggestions dynamiquement pour vos utilisateurs
- Les Pipes : Guide Complet - Apprendre à utiliser des pipes avec Angular 6

I - Démarrer un projet avec Angular-CLI (6.0.0).....	3
I-A - Au-delà du ng serve.....	3
I-B - Outillage.....	3
I-B-1 - NodeJS et Npm.....	3
I-B-2 - TypeScript.....	4
I-B-3 - Installer un IDE qui supporte TypeScript.....	4
I-C - Qu'est-ce que Angular-CLI ?.....	4
I-D - Installer l'outil Angular-CLI.....	5
I-E - Créer le socle de notre projet.....	6
I-F - Présentation du socle généré par Angular CLI.....	7
I-G - Quelques commandes utiles d'Angular CLI.....	10
II - Conclusion.....	10
III - Pour aller plus loin.....	11
IV - En résumé.....	11
V - Remerciements.....	11

I - Démarrer un projet avec Angular-CLI (6.0.0)

I-A - Au-delà du ng serve...

Imaginons la scène suivante, vous devez démarrer un nouveau projet plutôt conséquent. Peu importe la raison, vous souhaitez donner vie à votre dernière idée, ou votre chef de projet revient d'une longue réunion avec un cahier des charges prêt (moins amusant). Plusieurs zones sont prévues au sein de votre application : un espace visiteur avec une page d'accueil, une page *about me*, etc., et un espace membre protégé, uniquement accessible pour les utilisateurs authentifiés. Bref, tous les éléments attendus ont été définis, il ne vous reste plus qu'à allumer votre PC pour commencer le développement !

Heu... J'ai un projet qui va contenir plusieurs modules, des dizaines de composants, etc. Comment est-ce que je dois commencer ? Quels sont les éléments à créer en premier, et où faut-il les placer ?

C'est une très bonne chose de se demander comment mettre en place le socle de son application, **avant** de commencer les développements. Même si vous n'avez pas encore de réponses, au moins vous vous posez de bonnes questions ! En effet, partir sur une application avec de bonnes bases vous permettra d'assurer la suite de vos développements avec sérénité.

Cependant, avant de développer le socle de notre application Angular, aussi ambitieuse soit-elle, nous allons d'abord commencer par nous équiper un minimum. Bon, vous êtes assis devant votre poste de travail ? Vous attaquez un projet Angular de zéro ? Face à un dossier vide sur votre ordinateur ? Eh bien, c'est parti !

I-B - Outillage

Il y a un certain nombre d'éléments à installer sur votre machine, pour pouvoir faire du développement Angular. J'imagine que pour la plupart d'entre vous, vous avez déjà installé les éléments d'outillage que je vais présenter ci-dessous, mais je veux être sûr de n'oublier personne. Et puis un petit état des lieux avant de commencer, ça ne fait de mal à personne. Surtout pour vérifier que vous n'avez pas une version trop ancienne de ces outils !

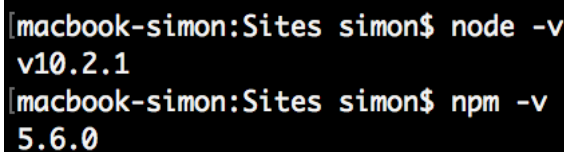
Voici donc les éléments à installer ou vérifier sur votre machine de développement :

I-B-1 - NodeJS et Npm

NodeJS et le Node Module Packager sont deux outils indispensables pour le développeur JavaScript moderne. L'installation de ces deux outils est possible depuis [cette page](#). Il s'agit d'un simple exécutable que vous téléchargez. Une fois installé, vous devez tester l'installation des différents éléments dans une console avec *node -v* et *npm -v* depuis un terminal.

Si vous êtes sous Windows, tapez les commandes *Windows + R*, puis *cmd*, puis la touche *Entrée*. Sur Mac, vous pouvez chercher le terme « *terminal* » avec Spotlight (*cmd + barre espace*). Et sous Linux, normalement, vous savez déjà !

Les versions requises pour développer avec Angular actuellement sont Node 8.x et Npm 5.x. Voici ce que j'obtiens sur mon poste de travail :



```
macbook-simon:Sites simon$ node -v
v10.2.1
macbook-simon:Sites simon$ npm -v
5.6.0
```

Figure 1 - Versions de Node et Npm

I-B-2 - TypeScript

Ce n'est plus un secret, Angular est écrit en TypeScript. Il s'agit du langage le plus recommandé, et quasi obligatoire, pour développer une application Angular. L'installation ou la mise à jour de TypeScript se fait à travers une commande unique :

```
npm install -g typescript@latest
```

Plus tard, dans le fichier de gestion des dépendances *package.json* de notre application, pensez à vérifier votre version de TypeScript. La version 2.5 au minimum est recommandée :

```
"typescript": "^2.7.2"
```

Figure 2 - Version de TypeScript

I-B-3 - Installer un IDE qui supporte TypeScript

Transition parfaite ! Vous venez d'installer TypeScript au niveau global sur votre machine. Il ne vous reste plus qu'à installer un outil de développement qui supporte le TypeScript.

Voici la liste des IDE recommandés sur le site de **TypeScript** :

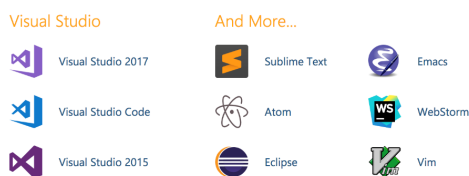


Figure 3 - Éditeurs de code recommandés par TypeScript

Je vous laisse utiliser celui qui vous correspond le mieux. Personnellement, j'utilise Atom pour sa polyvalence, mais il n'y a rien d'imposé à ce sujet.

I-C - Qu'est-ce que Angular-CLI ?

Pour générer le socle d'une application Angular, on fait souvent le même travail répétitif : ajouter le fichier de déclaration des dépendances *package.json*, création du fichier *index.html*, installation des dépendances avec la commande *npm install*, etc. C'est pratique pour apprendre au début, mais pour développer une application professionnelle, ce n'est pas ce qui est le plus recommandé.

En fait, l'équipe Angular de Google nous propose un outil en ligne de commande, qui nous permet d'automatiser les tâches de développement les plus courantes. Cet outil se nomme **Angular-CLI**. Voici ce que vous faisiez à la main jusqu'à maintenant, et que nous allons pouvoir automatiser :

- Générer l'architecture d'une nouvelle application : emplacement des dossiers, préremplissage des fichiers indispensables comme *index.html* ou *package.json*, par exemple.
- Générer de nouveaux composants, de nouveaux services, de nouveaux *pipes*, etc.
- Inspecter la syntaxe de notre code.
- Lancer des tests automatisés pour vérifier la qualité de notre code.
- Construire une archive de notre application, prête à déployer pour la production.

Techniquement, nous ne sommes pas obligés d'utiliser Angular-CLI pour développer notre application Angular, mais beaucoup de fonctionnalités embarquées par cet outil nous permettent d'améliorer fortement la qualité de notre code, et nous pouvons également économiser beaucoup de temps ! Voyons tout de suite comment installer ce formidable outil !

I-D - Installer l'outil Angular-CLI

Cependant, je vous déconseille de faire cela, car cela impose d'utiliser la même version d'Angular-CLI pour tous vos projets. Lorsque vous commencez à utiliser l'outil, cela ne pose pas de problème. Mais imaginez dans quelques années, si vous imposez une version d'Angular-CLI pour tous vos projets, y compris les plus anciens. Autant vous éviter de futurs problèmes de compatibilité dès maintenant.

I-E - Créer le socle de notre projet

Pour générer le socle de notre nouveau projet, nous allons utiliser la commande `ng new`. Il s'agit d'une commande d'Angular-CLI, qui vous permet de générer une architecture de fichiers et de dossiers robuste pour votre nouveau projet.

Heu... est-ce que la simple commande `ng new` est suffisante pour démarrer un projet professionnel ?

Non. Vous avez eu raison de poser la question. En fait, il existe plusieurs options que vous pouvez passer à la commande `ng new`, afin de paramétrer un minimum le socle de l'application qui sera générée. Voici les principales options que nous allons utiliser :

- L'option `--routing` : cette option vous permet d'ajouter un fichier de configuration pour les routes de votre application. Comme nous sommes fainéants, et que nous ne voulons pas rajouter ce fichier nous-mêmes par la suite, nous allons demander à Angular-CLI de le faire pour nous.
- L'option `--prefix` : cette option vous permet de préfixer le nom de vos composants par votre propre préfixe. Le préfixe par défaut est `app`. Par exemple, pour un composant `Home`, son nom sera donc `app-home`. Si vous utilisez votre propre préfixe, par exemple vos initiales, le nom sera `sd-home` (les initiales de « Simon Diény » dans mon cas). Pour votre préfixe, il est recommandé de choisir entre deux et cinq lettres, qui font référence au nom de votre application. Par exemple, le préfixe `todo`, ou `td`, est correct pour votre application de « Todo List ».
- L'option `--scss` : cette option vous permet de préciser le préprocesseur LESS pour votre application. Par défaut, l'option vaut « `css` », ce qui signifie que vous n'utilisez aucun préprocesseur dans votre projet, car vous utiliserez directement du code CSS. (Si vous ne savez pas ce qu'est un préprocesseur, ça n'est pas très important pour la suite du cours. Un préprocesseur vous permet de factoriser du code CSS afin de vous éviter des duplications dans vos feuilles de style. Mais comme le préprocesseur génère du CSS en sortie, vous pouvez continuer à utiliser du code CSS, tout en vous laissant la possibilité d'utiliser un préprocesseur plus tard dans votre vie de développeur).

Pour résumer les points précédents, voici donc la commande à exécuter pour démarrer un projet professionnel avec Angular :

```
ng new <nom_application> --prefix=sd --style=scss --routing
```

Ensuite, il va falloir que nous soyons patients, car Angular-CLI va générer l'architecture de toute notre application, et installer les dépendances avec `npm install` à notre place.

Une fois que Angular-CLI aura terminé, positionnez-vous à la racine de votre projet :

```
cd <nom_application>
```

Enfin, Angular-CLI nous fournit une deuxième commande pour démarrer notre projet, et ouvre un navigateur qui pointe sur notre application, tout cela automatiquement ! Exécutez donc la commande :

```
ng serve --open
```

Par défaut, votre application va se lancer sur le port 4200. Si vous préférez démarrer votre application sur un port différent, c'est possible avec l'option `port` !

```
ng serve --o --port=3000
```

L'option `--o` et `--open` sont équivalentes, il s'agit d'un raccourci de syntaxe, mais je préfère éviter les raccourcis dans les tutoriels, cela n'apporte pas grand-chose hormis de la confusion selon moi.

Patientez encore quelques instants (eh oui, encore !). Voici ce que vous devriez obtenir :

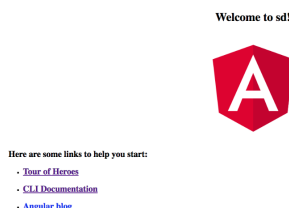


Figure 5 - Un « Hello, world ! » avec Angular-CLI

Angular-CLI a mis en place le socle de notre application, et a même lancé notre application dans un nouvel onglet !

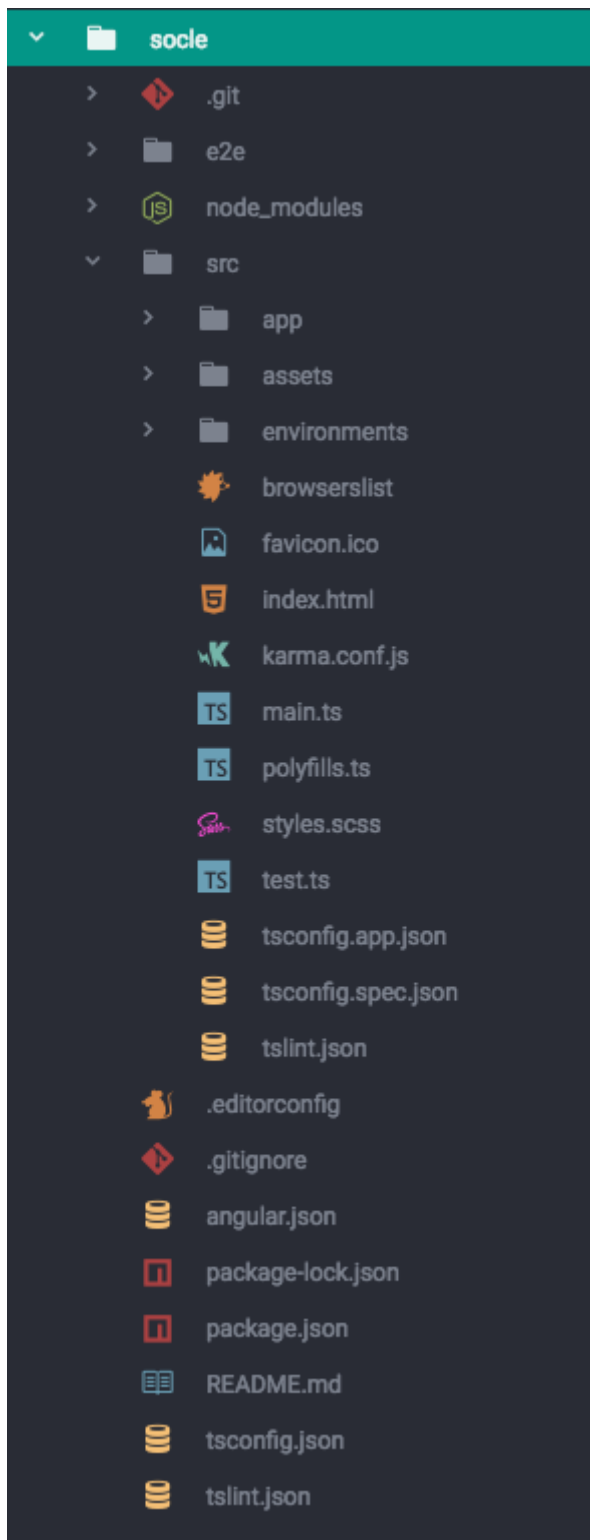
De plus, nous disposons d'une architecture qui contient un fichier de configuration de nos routes, un suffixe défini au niveau global de l'application, et qui utilise un préprocesseur CSS. Tout cela en une seule ligne de commande !

Malheureusement, beaucoup de personnes utilisent la commande `ng new` sans aucune option, et se retrouvent à devoir faire à la main ce qu'Angular-CLI aurait pu faire pour eux. En tant que développeur, on préfère déléguer le travail à Angular-CLI.

I-F - Présentation du socle généré par Angular CLI

Jusqu'à maintenant, tout ce que nous avons fait avait l'air... « magique ». Exécuter des commandes par ici, exécuter des commandes par-là, et hop, on obtient un socle prêt pour notre application Angular. Pourtant, il n'y a rien de magique. Je vous invite donc à ouvrir votre éditeur de texte favori, et à ouvrir le socle que Angular-CLI a généré pour nous. Et là, c'est la panique.

Mais... Qu'est-ce que tous ces fichiers font là ? Une application devrait pouvoir fonctionner avec moins de fichiers non, pourquoi ce superflu ?



Je vous propose de vous expliquer à quoi correspondent tous ces fichiers. Une fois que vous aurez compris, vous verrez que chaque fichier à sa place. Je vous propose donc de voir à quoi tous ces éléments correspondent :

- **Le dossier git** : Angular-CLI est livré avec le système de versioning Git par défaut. Vous pouvez donc vous en servir pour versionner votre projet. Si vous n'avez jamais entendu parler de Git, ou que vous ne l'aviez jamais utilisé, vous pouvez supprimer ce dossier. Votre application continuera à fonctionner parfaitement, promis !

- **Le dossier e2e** : ce dossier contient les tests *end-2-end* de votre application. Il s'agit de tests qui permettent de simuler des parcours dans votre application du point de vue de l'utilisateur, et s'assurer que tout va bien. Par exemple, « je me connecte », « je modifie mon identifiant », « je me déconnecte », etc. Si vous n'avez jamais entendu parler de ce genre de tests, vous pouvez ignorer ce dossier. Conservez-le quand même, au cas où vous souhaiteriez développer des tests plus tard pour votre application.
- **Le dossier node_modules** : il s'agit du dossier qui contient toutes les dépendances dont notre application a besoin pour fonctionner. Angular-CLI a déjà installé ces dépendances pour nous.
- **Le dossier src** : c'est dans ce dossier que se trouve l'essentiel du code source de notre projet. C'est dans ce dossier que vous allez passer la plupart de votre temps en tant que développeur. Tous nos composants Angular, nos templates, nos styles, et tous les autres éléments de notre application se trouveront dans ce dossier. Tous les fichiers en dehors de ce dossier sont destinés à soutenir la création de votre application. Vous retrouvez dans ce dossier plusieurs éléments :
 - **Le dossier app** : le dossier qui contient le code source de notre application, comme mentionné ci-dessus.
 - **Le dossier assets** : c'est le dossier qui contient les images de votre application. Vous pouvez également ajouter d'autres fichiers dont vous pourriez avoir besoin. Un fichier PDF à télécharger pour vos utilisateurs par exemple.
 - **Le dossier environments** : ce dossier contient un fichier de configuration pour chacun de vos environnements de destination : développement, production, etc. Vous pouvez définir des variables d'environnements pour chaque environnement, comme une URL de destination pour vos appels HTTP, qui sera certainement différente entre votre machine de développement et le serveur de production. Les fichiers seront remplacés à la volée lorsqu'on générera une archive de notre application.
 - **Le fichier browserlist** : c'est un fichier de configuration utilisé par Angular-CLI pour paramétrer certains outils en fonction de tel ou tel navigateur.
 - **Le fichier favicon.ico** : il s'agit de la fameuse icône qui s'affiche dans l'onglet de votre navigateur lorsque vous lancez votre application. Par défaut, il s'agit du logo d'Angular.
 - **Le fichier index.html** : c'est l'unique page HTML de notre application ! C'est elle qui contiendra l'ensemble de notre application.
 - **Le fichier karma.conf.js** : c'est le fichier de configuration de Karma, qui est un outil permettant d'exécuter des tests unitaires dans votre application.
 - **Le fichier main.ts** : c'est le point d'entrée principale de notre application. Il s'occupe de compiler notre application, et lance le module racine de celle-ci.
 - **Le fichier polyfill.ts** : les différents navigateurs existants n'ont pas le même niveau de prise en charge des normes du Web. Les polyfills servent à lisser ces différences en uniformisant le comportement de votre code entre tous les navigateurs.
 - **Le fichier style.scss** : le fichier qui contient le style global de votre application. La plupart du temps, vous aurez besoin que vos styles soient attachés à un composant. Cependant, pour une maintenance plus facile, les règles de style qui affectent toute votre application doivent être centralisées. Vous pouvez bien sûr ajouter du code CSS classique dans ce fichier si vous n'utilisez pas de préprocesseur.
 - **Le fichier test.ts** : c'est le point d'entrée principal pour vos tests unitaires.
 - **Le fichier tsconfig.app.json** : la configuration du compilateur TypeScript pour notre application Angular.
 - **Le fichier tsconfig.spec.json** : la configuration du compilateur TypeScript aussi, mais pour les tests unitaires cette fois.
 - **Le fichier tslint.json** : c'est un fichier de configuration qui définit une syntaxe de code commune à notre projet, et nous aide ainsi à garder un code cohérent.
 - **Le fichier editorconfig** : ce fichier permet de préciser quelques éléments pour votre éditeur de texte, comme le nombre d'espaces pour les tabulations. L'éternel débat (2 ou 4 espaces) n'est donc pas fini ! Je recommande deux indentations personnellement, et il s'agit du format généré par défaut par Angular CLI. Vous n'avez donc rien à changer dans ce fichier.
 - **Le fichier gitignore** : les fichiers à ignorer pour le versioning avec Git. Si vous n'utilisez pas Git, vous pouvez supprimer ce fichier. Sinon gardez-le, il est déjà préconfiguré pour le projet.
 - **Le fichier angular.json** : la configuration d'Angular-CLI. Vous pouvez définir plusieurs valeurs par défaut et configurer également les fichiers inclus lors de la création de votre projet. Par exemple, c'est dans ce fichier que se trouve le préfixe que l'on a défini précédemment en ligne de commande.
 - **Le fichier package-lock.json** : il s'agit d'un fichier propre au fonctionnement du *Node Package Manager*, qui a à voir avec les dépendances de vos dépendances. En effet, vous déclarez vos dépendances dans le fichier *package.json*, mais vos dépendances peuvent elles-mêmes avoir des dépendances qui sont en conflit. Je ne rentre pas plus dans les détails, car normalement, vous n'aurez jamais à modifier ce fichier. Dites-vous simplement que vous n'avez pas à vous en occuper.

- **Le fichier `package.json`** : le fichier bien connu qui vous permet de déclarer toutes les dépendances de votre projet. Vous pouvez également ajouter vos propres scripts personnalisés dans ce fichier, en plus de ceux d'Angular-CLI.
- **Le fichier `README.md`** : un fichier vous permettant de renseigner des informations pour démarrer et développer votre projet. Si vous êtes motivés, vous pourrez essayer de modifier ce fichier, afin d'indiquer à toute personne récupérant votre projet comment démarrer l'application sur sa machine !
- **Le fichier `tsconfig.json`** : c'est le fichier principal de la configuration de TypeScript, celui que j'ai mentionné au-dessus est simplement une configuration supplémentaire qui étend celle-ci.
- **Le fichier `tslint.json`** : configuration de peluches pour TSLint avec Codelyzer, utilisée lors de l'exécution de `ng lint`. Linting aide à garder votre style de code cohérent. De même, c'est le fichier principal de configuration, l'autre ne fait qu'étendre.

Bon, j'espère ne pas vous avoir endormis avec mes explications. Même si vous n'avez pas compris précisément le rôle de chacun des fichiers, reprenez simplement que les fichiers sources sont dans le dossier `src`, et que le reste concerne la configuration et les tests de votre application. C'est grandement suffisant pour commencer avec Angular-CLI !

Avant de terminer, j'aimerais également vous présenter quelques commandes mises à disposition par Angular-CLI, en plus de celles que nous venons de voir.

I-G - Quelques commandes utiles d'Angular CLI

L'air de rien, nous avons déjà utilisé deux commandes proposées par Angular-CLI : `ng new` et `ng serve`. Comme vous vous en doutez, Angular-CLI ne se limite pas à cela, et nous propose d'autres commandes.

Voici les commandes que vous utiliserez le plus souvent lors de vos développements :

- `ng generate` : cette commande permet de générer un nouvel élément dans votre application, avec un squelette déjà prêt, qui respecte toutes les bonnes pratiques recommandées par les équipes d'Angular. Ce nouvel élément peut être un composant, un module, un service, etc. Ensuite, vous n'avez « plus qu'à » compléter le squelette généré par Angular-CLI. Pratique !
- `ng lint` : cette commande inspecte la syntaxe de votre code, et affiche les erreurs relevées dans votre terminal de commande : oubli d'un point-virgule, des sauts de lignes inutiles dans vos méthodes, etc. N'hésitez pas à lancer cette commande régulièrement pour vous assurer que votre code reste propre.
- `ng test` : cette commande exécute les tests unitaires de votre projet. Comme vous n'en avez pas encore développés, cette commande exécutera simplement les tests générés par Angular-CLI.
- `ng e2e` : cette commande exécute également des tests, mais les tests « end-2-end » que je vous ai décrits précédemment.
- `ng doc` : cette commande prend simplement un terme de recherche en paramètre, et ouvre pour vous la documentation officielle d'Angular, angular.io, dans un nouvel onglet correspondant au mot recherché. Cela vous sera utile si vous ne savez plus comment utiliser tel service ou tel filtre d'Angular. Imaginons que vous souhaitiez afficher du texte en minuscule. Pour cela, exécutez la commande : `ng doc lowercase`. Un nouvel onglet de votre navigateur s'ouvrira alors, avec les informations concernant l'utilisation du filtre `lowercase` d'Angular.

Les commandes `ng new` et `ng serve` sont bien sûr les premières commandes que vous utiliserez, car elles sont très pratiques pour démarrer votre projet. Angular-CLI propose ensuite d'autres commandes, que je viens de vous présenter. Celles-ci vous serviront tout le long de votre processus de développement, et nous verrons dans un prochain tutoriel comment générer une architecture technique professionnelle pour votre application Angular, grâce à ces commandes justement. Nous gagnerons ainsi beaucoup de temps !

II - Conclusion

Voilà, comme nous avons pu le voir dans ce tutoriel, il est possible d'économiser beaucoup de temps de développement avec Angular-CLI, et vous auriez tort de vous en priver.

Je ne peux que vous encourager à utiliser Angular-CLI pour vos prochains développements, c'est un outil fiable maintenu par les équipes de chez Google, et incroyablement pratique.

Vous connaissez maintenant plusieurs commandes, dont la commande à exécuter pour générer le socle de votre prochain projet, au-delà du simple `ng serve` que l'on peut trouver sur la documentation officielle.

Enfin, vous devriez avoir une meilleure connaissance des différents éléments générés par Angular-CLI. Vous pouvez commencer vos développements sur une bonne base désormais.



III - Pour aller plus loin

Si vous souhaitez recevoir les prochains articles dédiés aux développements d'applications professionnelles avec Angular, je vous invite à vous inscrire à cette [liste email](#). Vous recevrez les prochains articles directement dans votre boîte mail, pour être sûr de ne rien rater.

IV - En résumé

- Angular-CLI est un outil permettant de générer un socle robuste pour créer votre prochaine application Angular.
- Angular-CLI fournit également plusieurs commandes pratiques permettant de générer de nouveaux éléments dans votre application, d'analyser la syntaxe de votre code, ou encore d'exécuter des tests automatisés.
- La commande `ng new` peut être agrémentée de plusieurs paramètres, afin de générer un socle plus complet pour les besoins d'une application professionnelle.
- La commande `ng serve` peut être utilisée pour exécuter notre application dans un navigateur.
- Le socle d'une application généré par Angular-CLI est composé de beaucoup de fichiers, et cela peut être déstabilisant la première fois. Retenez simplement que votre code source se trouve dans le dossier `src`, et que le reste des fichiers constitue des éléments de configuration et de tests.
- Il est recommandé d'utiliser Angular-CLI pour démarrer un projet avec Angular, car il vous permet de mettre en place le socle de votre application rapidement, et vous fait économiser du temps de développement par la suite grâce aux commandes qu'il met à disposition.

V - Remerciements

Je tiens à remercier  **Marco46** pour la relecture technique, et  **f-leb** pour la relecture orthographique. Ils ont permis à l'article que vous lisez aujourd'hui de voir le jour.