Cours Angular

1.0 | Fondations

Pré- requis (**rappels**), installation du framework et première application et ajustements

1] Outils nécessaires

L'installation du cadre de développement pour Angular dépend du système utilisé. Mais, une fois installé, son utilisation sera identique car spécifique aux outils et non aux systèmes.

- ***Windows (recommandé → version 10 - 64 bits)***

1) L'IDE **Visual Studio Code** : https://code.visualstudio.com/download

→ Choisir `System Installer`, version 64 bits

2) Le framework **NodeJS** : https://nodejs.org/en/download

→ Choisir `Windows Installer (.msi)`, version 64 bits

3) Le navigateur **Google Chrome** : https://www.google.com/intl/fr_fr/chrome

→ L'installateur propose automatiquement la bonne version

4) Le logiciel de gestion de versions **Git** : https://git-scm.com

→ Optionnel mais recommandé, l'installateur propose automatiquement la bonne version
→ Positionnez bien sûr obligatoirement les variables Git `user.name` et `user.email`

- ***Linux (recommandé → distribution Ubuntu 20.04 LTS - 64 bits)*** *en mode* ***administrateur***

1) L'IDE **Visual Studio Code** :

```
# curl https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > microsoft.gpg
# install -o root -g root -m 644 microsoft.gpg /etc/apt/trusted.gpg.d/
# echo "deb https://packages.microsoft.com/repos/vscode stable main" >
 /etc/apt/sources.list.d/vscode.list
# rm microsoft.gpg
# apt install code
```

Les projets en JavaScript nécessitant un nombre élevé de fichiers, il est nécessaire d'augmenter cette limite sur le système. Il faut donc ajouter à la fin du fichier `/etc/sysctl.conf` la ligne `fs.inotify.max_user_watches=524288`, puis toujours en root, taper la commande : `sysctl -p –system`.

2) Le framework **NodeJS** :

→ Ne pas prendre la version des dépôts, souvent obsolète ; actuellement on se base sur la version LTS 16.*

```
# curl -sL https://deb.nodesource.com/setup_16.x | bash -
# apt install nodejs
```

3) Le navigateur **Google Chrome** :

→ Ne pas prendre la version du site

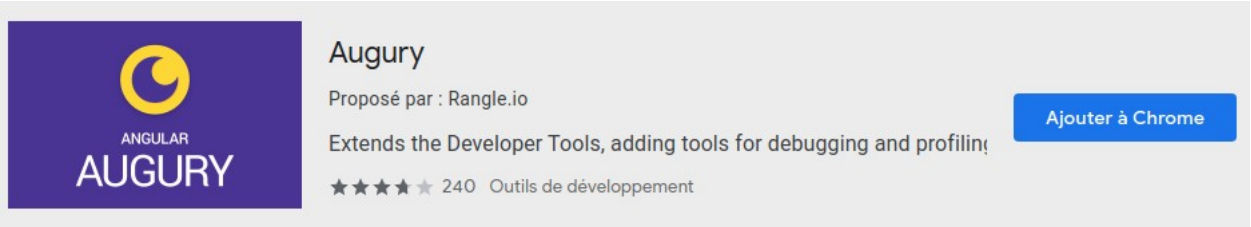
```
# wget -q -O - https://dl-ssl.google.com/linux/linux_signing_key.pub | apt-key add -
# echo "deb http://dl.google.com/linux/chrome/deb/ stable main" >
 /etc/apt/sources.list.d/google-chrome.list
# apt update
# apt install google-chrome-stable
```

4) Le logiciel de gestion de versions **Git** :

→ Installer la version des dépôts par `apt install git`
→ Positionnez bien sûr obligatoirement les variables Git `user.name` et `user.email`

- ***Utilisation de l'extension pour Chrome***

L'extension Chrome pour Angular au niveau des outils de développements se nomme **Augury** dans le Web Store :



2] Création d'une application

Comme le lancement de l'application se fait par le serveur Web intégré d'Angular, le placement du dossier de l'application n'a pas d'importance.

- ***Installation d'Angular (une seule fois)***

T2.1 Ouvrez une console avec `Node.js Command Prompt` sur Windows (impératif pour avoir l'environnement) ou un terminal en root sous Linux pour tapez la commande :

```
# npm -g install @angular/cli
# ng version
```

Cette commande installe pour le système (option `-g`) l'outil d'interface en ligne de commande nécessaire à la création d'applications Angular. La deuxième commande affiche la version d'Angular.

- ***Création proprement dite d'une application***

T2.2 Dans la même console (mais cette fois un terminal utilisateur sous Linux), à l'endroit désiré (le plus « propre » serait dans un dossier spécifique aux projets Angular).

```
$ mkdir angular
$ cd angular
```

Pour créer un projet vous avez le choix de mettre immédiatement dans la ligne de commande deux options. Si vous ne le faites pas, ces deux options vous seront demandées.

T2.3 Créez le projet et lancez-le avec les options :

```
angular$ ng new test --style=css --routing=false
```

Les options indiquent respectivement l'utilisation classique des feuilles de style et la non création d'un fichier de configuration pour les routes de l'application.

Voir la documentation pour avoir les différentes options de la commande `ng new` :
https://angular.io/cli/new.

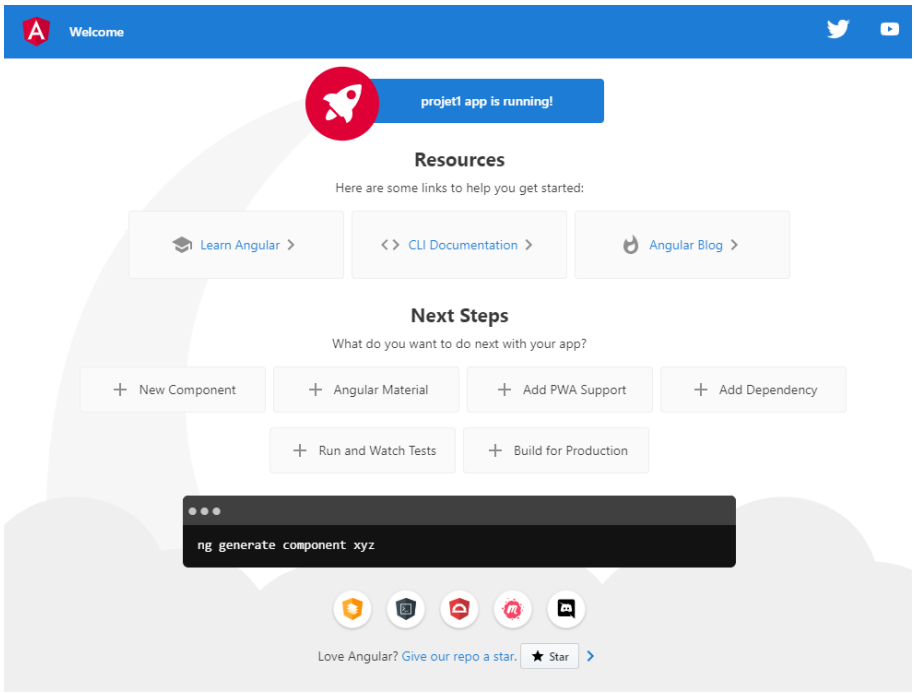
Pour la première option, vous trouverez plus de renseignements ici :
https://www.alsacreations.com/article/lire/1717-les-preprocesseurs-css-c-est-sensass.html.

Le routage (deuxième option) permettra plus tard à Angular de naviguer entre les différentes pages de l'application à l'aide du module `RouterModule`.

T2.4 Allez dans le répertoire du projet (oubli fréquent du débutant) et lancez l'application par le serveur intégré d'Angular :

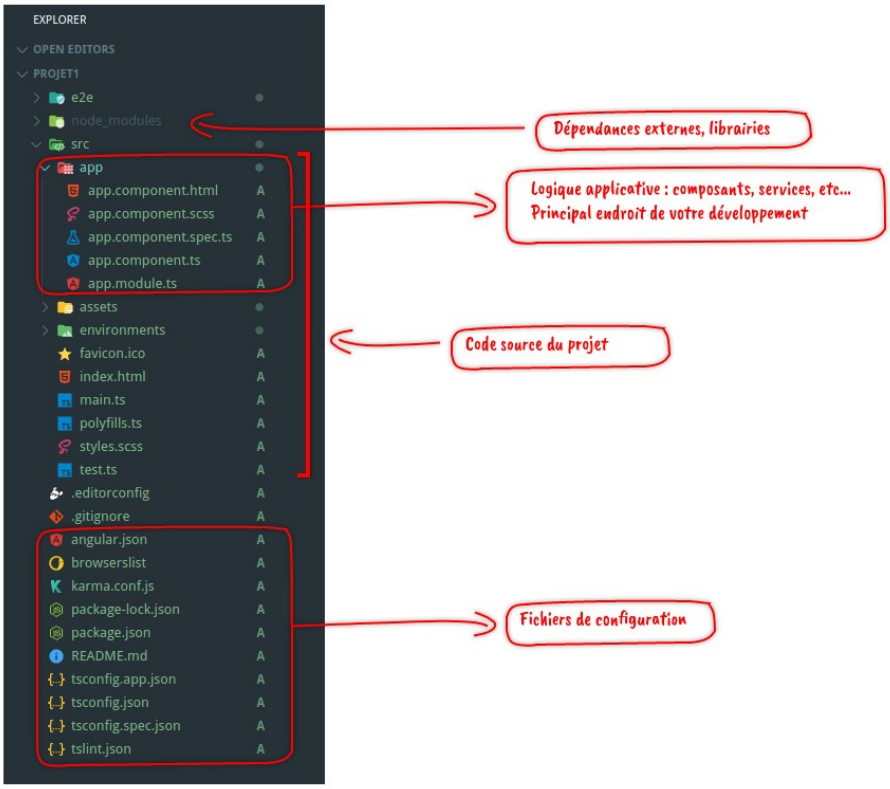
```
angular$ cd test
test$ ng serve
```

Cette console ne devra pas être fermée ; on se retrouve avec le site Web classique de démarrage à l'adresse `localhost:4200` dans le navigateur :



- ***Structure de l'application***

T2.5 Ouvrez l'IDE sur le dossier du projet afin de bien voir la logique de structure d'Angular, à savoir le fichier `app.component.ts` contenant la logique d'injection du composant (sélecteur `app-root`) à partir du module principal `app.module.ts` :



3] Ajustements

NodeJS, plate-forme de base pour le développement de l'application Angular, comporte des versions de librairies (dépendances ou *packages*) gérées pas NPM avec une mise à jour particulière (pour ne pas dire délicate…).

- ***Le fichier `package.json`***

Regardons les dépendances indiquées dans ce fichier (extrait) :

```
"dependencies": {
  "@angular/animations": "~12.2.0",
  "@angular/common": "~12.2.0",
  "@angular/compiler": "~12.2.0",
  "@angular/core": "~12.2.0",
  "@angular/forms": "~12.2.0",
  "@angular/platform-browser": "~12.2.0",
  "@angular/platform-browser-dynamic": "~12.2.0",
  "@angular/router": "~12.2.0",
  "rxjs": "~6.6.0",
  "tslib": "^2.3.0",
  "zone.js": "~0.11.4"
},
```

Le symbole (ou son absence) devant le numéro de version signifie :

`version` → une correspondance exacte au numéro de version
`~version` → une version « approximativement » équivalente
`^version` → une version compatible avec ce qui est demandé

- ***Les mises à jour éventuelles***

T3.1 Tapez la commande `npm` pour la vérification des packages :

```
test$ npm audit
```

Le résultat de cette commande fait ressortir des vulnérabilités, certaines moyennes, d'autres sévères ! Pas de panique… l'essentiel porte sur des dépendances en mode développement. Le plus important est le résultat de cette commande :

```
test$ npm audit --production
```

Il ne devrait pas y avoir de vulnérabilités...

T3.2 Pour être un peu plus complet, on peut voir le résultat de la commande ci -dessous :

```
test$ npm outdated
```

```
Package          Current    Wanted   Latest  Location                    Depended by
@types/jasmine     3.8.2     3.8.2   3.10.1  node_modules/@types/jasmine  test
@types/node     12.20.36  12.20.36  16.11.6  node_modules/@types/node     test
jasmine-core       3.8.0     3.8.0   3.10.1  node_modules/jasmine-core    test
rxjs               6.6.7     6.6.7    7.4.0  node_modules/rxjs            test
typescript         4.3.5     4.3.5    4.4.4  node_modules/typescript      test
```

Ce qui importe c'est que la version demandée (`Wanted`) soit la même que celle courante. C'est le cas ici, il n'y a donc rien à faire. Dans le cas contraire, on supprime le fichier `package-lock.json`, on indique le numéro demandé dans le fichier `package.json` pour le paquet incriminé (préférez la correspondance exacte) et on tape la commande `npm install`.