Angular

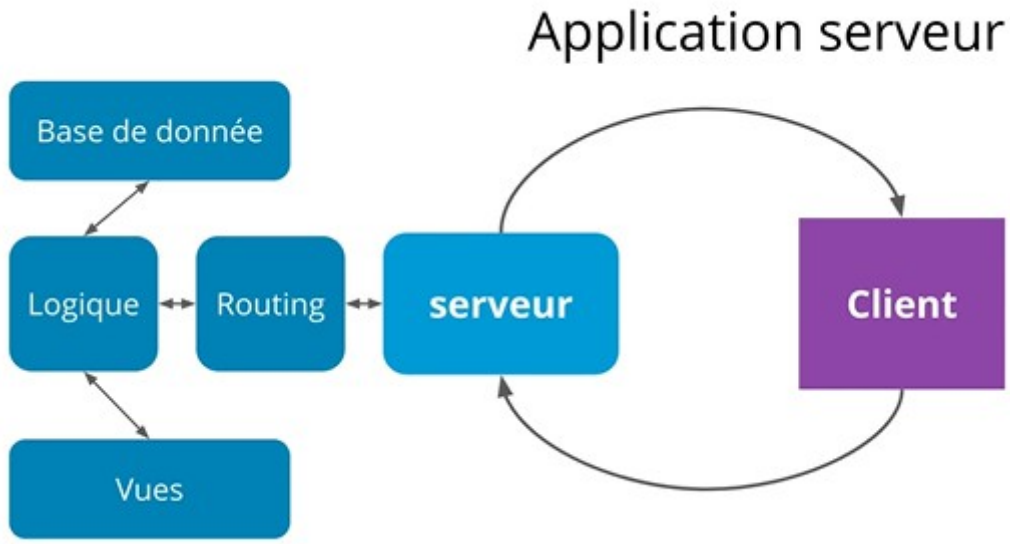
Qu'est ce qu'une single page application ?

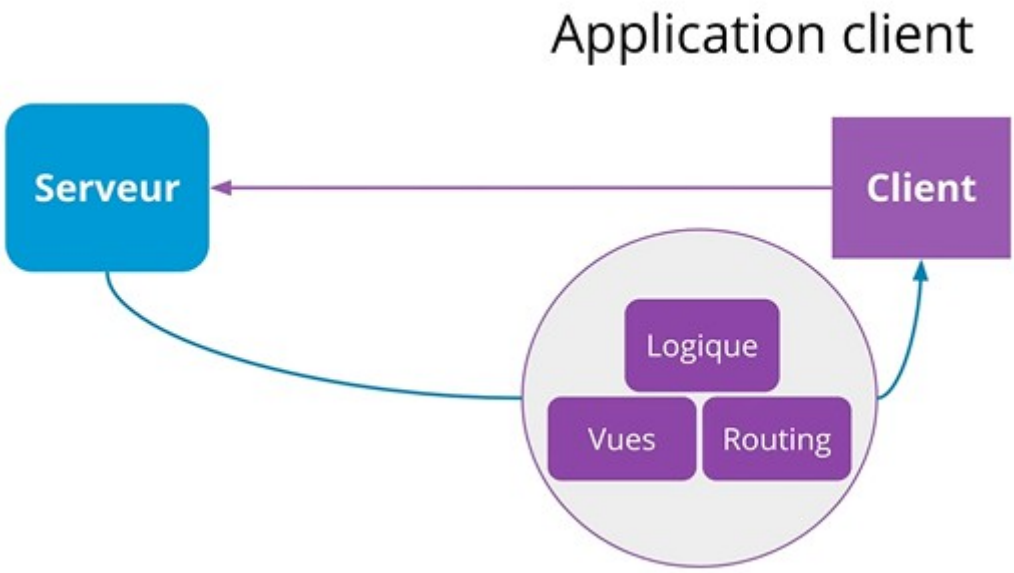
Angular est un framework pour créer des Single Pages Applications (SPA).

Une Single Page Application (SPA) est une application qui fonctionne dans un navigateur sans que l'utilisateur n'ait besoin de recharger la page.

Le principe est de se rapprocher d'une application hors ligne : pas de rechargement de pages, de la rapidité / fluidité, pas d'attente supplémentaire due à un réseau etc.

Application serveur
Application client → créer à l'aide d'Angular.
Elle possède un temps initial de chargement qui est relativement long mais ne se fait qu'une seule fois.





Les principales caractéristiques de la SPA sont :

- - le rendu est effectué côté client (quand un élément change, la page est modifiée grâce aux scripts de l'application chargée côté client)
- - pour fonctionner elle charge en principe une seule fois l'application (HTML, CSS et scripts)
- - seules les données sont transmises, si nécessaire, entre le serveur et l'application client (le plus souvent au format JSON)
- - le développement mobile est simplifié car le code backend peut être utilisé que l'application soit Web ou native (iOS, Android)
- - elle est particulièrement adaptée pour stocker les données localement et n'envoyer des requêtes au serveur que lorsque c'est nécessaire

Les désavantages d'une SPA sont :

- - le chargement initial de l'application prend du temps
- - les robots des moteurs de recherche n'attendront pas le chargement et le SEO en sera impacté

Mais ces problèmes peuvent être résolus de plusieurs manières, notamment avec le Server Side Rendering, comme nous le détaillerons dans le cours.

Installation d'Angular CLI et création de notre première application

Installation de `Prettier`

`Prettier` permet de mettre en forme le code automatiquement lorsque vous le sauvegardez.

C'est une extension très utile et nous vous invitons à l'installer dans `VS code`;

Allez dans `Extensions` puis recherchez `Prettier`. Ensuite cliquez sur `Install`.

`Prettier` a plus de 14 millions de téléchargement sur `VS code` ce qui en fait l'une des extensions les plus utilisées.

Dans `VS Code` allez ensuite dans `File` puis `Preferences` puis `Settings`. Recherchez `format on save` et cochez la case.

Toujours dans `VS Code`, et toujours dans `Settings`. Recherchez cette fois `Default Formatter` et dans la liste déroulante sélectionnez `Prettier - Code Formatter esbenp.prettier-vscode`.

`Prettier` fonctionnera maintenant à chaque fois que vous sauvegardez !

Angular CLI

Il s'agit d'un outil développé par l'équipe de Google afin d'initialiser, de développer et de maintenir des applications Angular.

Pour l'installer :

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

Pour vérifier l'installation :

```
ng version
```

3) Création d'un projet

Etape 1 : création d'un nouveau projet

Entrez la commande

```
ng new monProjet
N // puis entrée
SCSS // puis entrée
code monProjet
```

Si vous faites `ng v` dans un projet, vous obtiendrez la version du CLI mais également de tous les principaux paquets.

Etape 2 : démarrer l'application

Pour construire l'application et démarrer un serveur web en local il suffit de taper dans le répertoire du projet :

```
ng serve
```

Ouvrez votre navigateur à l'adresse suivante :

```
http://localhost:4200
```

Angular CLI a lancé un serveur de développement Node qui va vous permettre d'accéder à votre application en local pendant vos développements.

Première approche d'un composant Angular :

Les applications Angular sont faites avec des composants (*components*). Un composant est une combinaison entre un template HTML et une classe composant qui contrôle une portion de l'écran.

1) app.component.ts

```
1   import { Component } from '@angular/core';
2
3   @Component({
4     selector: 'app-root',
5     templateUrl: './app.component.html',
6     styleUrls: ['./app.component.css']
7   })
8   export class AppComponent {
9     title = 'test';
10  }
11
```

Chaque composant contient un décorateur `@Component` qui prend un objet contenant des métadonnées. Elles contiennent l'emplacement du template HTML et du style pour le composant.

La propriété `selector` permet d'indiquer à Angular d'afficher le composant à l'emplacement d'une balise html spécifique, ici `<app-root>` dans le fichier `index.html`.

2) index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Test</title>
    <base href="/" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="icon" type="image/x-icon" href="favicon.ico" />
  </head>
  <body>
    <app-root></app-root> <!-- Indique à Angular où afficher le composant app-root -->
  </body>
</html>
```

Nous retrouvons bien la balise `<app-root>` dans le fichier `index.html` qui indique où doit être placé le composant.

3) app.component.html

Le composant contient tout le HTML nécessaire pour l'écran de présentation de l'application d'exemple. Vous pouvez supprimer son contenu dès maintenant et le remplacer par `{{ title }}`.

`{{ title }}`, est une interpolation. Lors de l'exécution, Angular remplace `{{ title }}` par la valeur de la variable `title` contenue dans la classe du fichier `app.component.ts`.

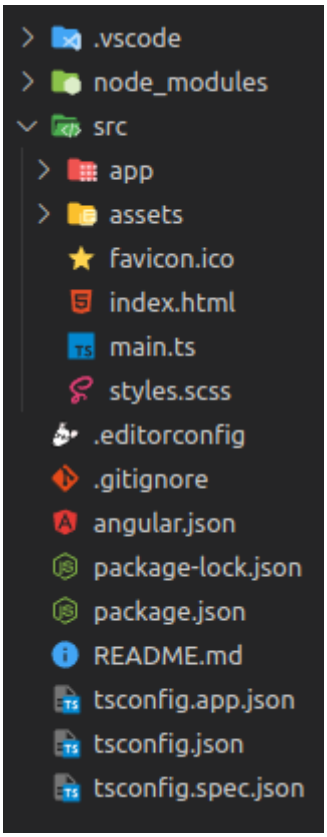
Bien sûr nous étudierons tout cela en détails dans les leçons à venir mais il s'agit d'un premier bref aperçu !

Sans décorateur, Angular ne peut pas savoir que la classe est un composant. Il ne s'agit pour lui que d'une classe standard.

Un composant Angular est la brique de base pour bâtir une application.

Comprendre la structure

L’arborescence



1) Racine ./

.editorconfig : Configuration simple pour votre éditeur pour vous assurer que tous ceux qui utilisent votre projet aient la même configuration de base (par exemple l'indentation, on vous laisse régler ça entre amis). La plupart des éditeurs prennent en charge ce fichier.

.gitignore : fichier git pour vous assurer que les fichiers générés automatiquement ne sont pas pris en compte. Par exemple les dépendances (/node_modules), ainsi que le build du projet (/dist).

angular.json : Configuration pour Angular CLI. Vous pouvez par exemple configurer le nom et l'emplacement du folder où sera buildé votre projet (`"outputPath": "dist/monProjet"` par défaut).

package.json : Configuration npm listant les paquets tiers que votre projet utilise (les dépendances). Vous pouvez également ajouter vos propres scripts personnalisés ici.

package-lock.json : Arbre exacte des dépendances et de leurs propres dépendances, permettant de réinstaller exactement les mêmes versions dans votre équipe.

README.md : Documentation de base pour votre projet, pré-remplie d'informations de commande CLI. Assurez-vous de l'améliorer avec la documentation du projet afin que quiconque puisse builder votre application en suivant les instructions contenues dans ce fichier !

tsconfig.app.json : Extension de la configuration du compilateur TypeScript pour compiler votre application.

tsconfig.json : Configuration du compilateur TypeScript pour compiler votre application. Ce fichier est également utilisé par votre IDE afin de vous donner des informations utiles (mauvais type d'input par exemples).

tsconfig.spec.json : Extension de la configuration du compilateur TypeScript pour compiler les tests pour votre application.

2) node_modules

Ce sont les dépendances installées par npm qui correspondent à celles déclarées dans `package.json`.

3) src

app/app.component.(ts, html, scss, spec.ts) : Définit AppComponent avec un template HTML (.html), une feuille de style SCSS ou CSS (.scss ou .css) et un test unitaire (.spec.ts). C'est le composant racine de ce qui deviendra un arbre de composants imbriqués à mesure que l'application évolue.

app/app.module.ts : Définit AppModule, le module racine qui indique à Angular comment assembler l'application. A l'initialisation du projet il déclare uniquement l'AppComponent. Dans la suite du cours, il y aura évidemment plus de composants déclarés.

assets/* : Un dossier où vous pouvez mettre des images qui seront copiées directement lorsque vous construirez votre application.

favicon.ico : comme son nom l'indique, permet de configurer le favicon de votre application pour les navigateurs.

index.html : La page HTML principale qui est servie en premier lorsque quelqu'un visite votre site. La plupart du temps, vous n'aurez jamais besoin de le modifier. Le CLI ajoute automatiquement tous les fichiers JS et CSS lors de la création de votre application afin que vous n'ayez jamais besoin d'ajouter manuellement des balises 'script' ou 'link'.

main.ts : Le point d'entrée principal de votre application. Compile l'application avec le compilateur JIT et bootstrap le module racine de l'application (AppModule) pour s'exécuter dans le navigateur.

styles.css (ou styles.scss) : Vos styles globaux vont ici. La plupart du temps, vous voudrez avoir des styles locaux dans vos composants pour en faciliter la maintenabilité, mais les styles qui affectent l'ensemble de votre application doivent être dans ce fichier.

TypeScript

https://dyma.fr/developer/list/chapters/core/58a1875106e2b482e9db7bb3/lesson/angular/58cd313ce748376f9610ddcb/1/7