IMT - cours Angular - A2

Prérequis:

git & node (14 ou +) installé sur votre machine.

Lab 1: ma première application Angular

Récupérez et ouvrez le fichier index.html fournie par votre professeur. Il contient le code html nécessaire pour l'affichage d'un site de ecommerce.

Votre objectif va être d'intégrer ce code dans une application Angular, puis de le découper en plusieurs composants distincts.

Etape 1: générer une application Angular.

Installez le CLI Angular sur votre machine.

Lancez la commande

```
ng version
```

pour vérifier que le CLI est correctement installé et que vous utilisez bien la dernière version (12).

Une fois le CLI installé, utilisez le pour générer une nouvelle application Angular (choisissez librement le nom et les options que vous préférez, cela n'aura pas d'impact sur la suite des exercices).

Prenez le temps d'ouvrir le dossier généré et de vous familiarisez avec votre nouvelle application.

Enfin, lancez le script de démarrage de votre application et rendez-vous sur l'url indiquée.

Etape 2: intégrer du code html

En utilisant le CLI, votre application arrive avec un code par défaut. Nous allons remplacer ce code par défaut par le code de la page ecommerce fournie par votre professeur.

Le premier composant de votre application est appelé "AppComponent".

Ouvrez le fichier contenant le template de ce composant et remplacé celui-ci par tout le code html de la page ecommerce fournie (tout ce qui se trouve entre les balises "body").

Assurez vous dans votre navigateur que l'affichage de votre application a changé.

A ce stade, même si l'affichage a changé, votre application ne ressemble plus à grand chose ... En effet, il manque du code css ! Le code de la page ecommerce s'appuit sur "Bootstrap" et quelques lignes de css globale mais ni l'un ni l'autre ne sont présents dans votre nouvelle application Angular.

Pour palier à ce problème, récupérer le lien vers Bootstrap et les lignes css présent au sein de la balise <head> du code fournie par votre professeur, et placez le dans le fichier index.html de votre application.

Vérifiez que l'affichage de votre application s'est amélioré.

Etape 3: découper votre application en composant

Votre application a beau être fonctionnel dès maintenant, on peut deviner qu'il sera difficile de maintenir le code de celle-ci car pour l'instant tout le code html de celle-ci se trouve à un seul endroit: le composant app. C'est une mauvaise pratique.

Vous allez donc découpez votre application en plusieurs composants. A savoir:

- un composant "NavbarComponent" en charge de l'affichage de la navbar
- un composant "FooterComponent" en charge de l'affichage du footer

Rappel: vous pouvez générer le code d'un composant à l'aide d'une commande du CLI.

Lab 2 interpolation & binding

Etape 1: interpolation

Dans votre composant app, créer (ou modifier si elle existe déjà) une variable "title" contenant la chaîne de caractère "IMT ecommerce". Utilisez l'interpolation pour afficher le contenu de cette variable dans le titre (h1) de votre application.

Créer dans votre projet un répertoire "model". A l'intérieur, créez un fichier "product.ts". Dans celui-ci, définissez une classe du même nom, avec comme propriété:

- title (string)
- photo (string)
- description (string)
- price (number)
- stock (number)

Dans votre composant "AppComponent", définissez une variable "products" qui contiendra une liste d'objet de type "Product". Placez dans cette liste toutes les données des produits défini en dur dans votre template (placez le stock que vous souhaitez pour chaque produit).

Afficher dans votre template le titre, le prix, la description et la photo de chacun de vos produits en vous basant uniquement sur votre variable "products".

Etape 2: property binding

Dans votre composant "AppComponent", utilisez le property binding pour désactiver le bouton "Ajouter au panier" de chacun de vos produits si celui-ci a un stock égal à 0.

Etape 3: event binding

Dans votre composant "AppComponent", utilisez l'event binding pour appelez une fonction de votre composant "addProductToBasket" (à définir) qui reçoit en paramètre le produit cliqué et qui fait un console.log de celui-ci.

Lab 3: communication entre composant.

Etape 1: un peu de refacto

Créer un composant "ProductComponent" dont le rôle sera d'afficher le template d'un produit.

Transférez dans ce composant:

- la portion de template html permettant d'afficher un produit
- la méthode addProductToBasket

Définissez une propriété "data" dans ce composant, de type "Product".

Faites en sorte qu'un composant parent puisse fournir une valeur à cette propriété "data", de la manière suivante:

```
<app-product [data]="product"></app-product>
```

Etape 2: gestion d'un panier

Jusqu'à maintenant, au clique sur un bouton "Ajoutez au panier", la seule chose que nous faisons est de logguer dans la console le produit cliqué". Nous allons maintenant faire en sorte que le produit soit ajouté dans un panier et que le total soit mis à jour correctement.

Dans votre composant Product, créer une nouvelle propriété "addToBasket" de type "EventEmitter" et assurez vous qu'il soit accessible depuis l'extérieur du composant.

Modifier votre composant pour que, au lieu de logguer le produit cliqué, l'évènement "addToBasket" soit émis, avec en paramètre les données du produit cliqué.

Dans votre composant "AppComponent", définissez une variable "basket" de type tableau de "Product".

Toujours dans le composant "AppComponent", créez une nouvelle méthode "addProductToBasket" qui sera déclenchée à chaque évènement "addToBasket" émis et qui ajoutera le produit dans votre panier.

Définissez une fonction "getTotal" en charge de calculer le montant total accumulé dans votre panier.

Utilisez cette fonction pour afficher le total dans votre application.

Lab 4: directives

Etape 1: ngIf

Jusqu'à maintenant, si un produit n'est plus en stock, il est malgré tout encore visible. Ce n'est pas une expérience utilisateur très efficace.

Au lieu de désactiver le bouton "Ajouter au panier", utilisez la directive "ngIf" pour supprimer de l'application le template d'un produit qui n'est plus en stock.

Etape 2: ngClass

Pour améliorer l'expérience utilisateur, on va indiquer explicitement quand un produit n'a plus qu'un seul exemplaire en stock.

Pour cela, utilisez la directive "ngClass" pour ajoutez une class css "last" sur la première balise du composant "ProductComponent" quand il n'y a plus qu'un seul produit en stock.

Dans le fichier de style du composant, ajoutez la définition suivante:

```
.last {
  background-color: rgba(255, 0, 0, 0.4);
}
```

Etape 3: ngFor

Dans le template de notre composant App, nous avons placé 4 fois la balise du composant "ProductComponent". Même si cela fonctionne, ce n'est pas quelque chose de maintenable (imaginez avoir 35 produits différents ...).

Utilisez la directive "ngFor" pour répéter l'utilisation du ProductComponent autant de fois qu'il existe de produits.

Lab 5: intégrer une bibliothèque de composants graphiques

Ajouter la bibliothèque "@angular/material" à votre application (vous êtes libre de choisir le thème que vous préférez).

Remplacer tous les boutons "Ajouter au panier" de votre application par des boutons "material" (n'hésitez pas à aller consulter les exemples de la documentation officielle).

Lab 6: pipe & service

Etape 1: pipe currency

Utilisez le pipe currency pour afficher le montant total de votre panier ainsi que les prix de vos produits avec le symbole € sans avoir besoin de le placer manuellement dans votre template.

Sentez vous libre d'intégrer tous les éléments material que vous voudrez dans l'application 😄

Etape 2: création de service

A l'heure actuelle, l'ensemble de la gestion des produits (la liste des produits, le calcul du total, la gestion du panier) se fait dans le composant AppComponent. C'est trop de responsabilités à un seul endroit et il est fort probable qu'au fur et à mesure que notre application grossit, nous ayons besoin de certaines de ces fonctionnalités dans d'autres composants.

Vous allez extraire ces fonctionnalités dans des services, et faire appel à ces services dans vos composants.

Créer un service "ProductService". Placez y:

- votre variable "products"
- une méthode "decreaseStock" qui prends en paramètre un produit et diminue d'un son stock.
- une méthode "isAvailable" renvoyant un boolean à vrai si le produit a encore du stock.
- une méthode "isLast" renvoyant un boolean à vrai si le produit n'a plus qu'un élément en stock.

Injecter ce service dans le composant App et utilisez le. A la fin de cette étape, votre application doit présenter les mêmes fonctionnalités qu'avant.

Créer un service "BasketService". Placez y:

- votre variable "basket"
- une méthode "addProduct" qui prendre en paramètre un produit et l'ajoute au panier.
- une méthode "getTotal" qui renvoie le total du panier.

Injecter ce service dans le composant App et utilisez le. Là encore, à la fin de cette étape votre application doit présenter les mêmes fonctionnalités qu'avant.