valtech_

Labs Angular

Labs Angular

Nous verrons au cours des travaux pratiques les points suivants :

Objectifs

Le labs a pour objectif de vous présenter les différentes fonctionnalités d'Angular. Au programme :

- Mise en place de l'environnement avec `@angular/cli`
- Création d'un composant chronomètre et d'un Pipe,
- Utilisation des services Angular pour communiquer avec le backend,
- Création d'un formulaire de saisie,
- Mise en place du routage.

TP 1 - Installation de l'environnement

Prérequis

Vérifier que vous avez les éléments suivants d'installé sur votre poste :

- Node v6 ou plus avec la commande npm –v,
- Git, nous l'utiliserons pour récupérer le projet initial,
- Webstorm (ou un autre IDE)

Installation

Initialisation du projet

Nous allons utiliser l'outil `@angular/cli` pour générer le projet. Placez-vous dans votre workspace et lancez les commandes suivantes :

```
npm install -g @angular/cli
ng new tp-angular
cd tp-angular
npm install --save @angular/material
```

> Une fois le projet créé vous pouvez l'ouvrir dans votre IDE.

Pour lancer le serveur de développement :

```
ng serve
```

Installation d'Angular Material

Nous allons utiliser Angular material pour créer les interfaces de nos applications.

Lancer la commande suivante:

```
npm install --save @angular/material
```

Puis rajouter dans le fichier `app.module.ts` la dépendence du module comme suivant :

```
import { MaterialModule } from '@angular/material';
// other imports
@NgModule({
  imports: [MaterialModule],
  ...
})
export class AppModule { }
```

En complément vous pouvez rajouter la feuille de style suivante dans la page index si vous voulez utiliser les icônes de la librairie Material :

```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
```

Générer des Composants, Directives, Pipe, etc...

Vous pouvez utiliser la commande `ng generate` (ou `ng g`) pour générer des composants Angular :

```
ng generate component my-new-component
ng g component my-new-component # using the alias

# components support relative path generation
# if in the directory src/app/feature/ and you run
ng g component new-cmp
# your component will be generated in src/app/feature/new-cmp
# but if you were to run
```

```
ng g component ../newer-cmp
# your component will be generated in src/app/newer-cmp
```

Voici la liste des commandes possibles pour générer une fonctionnalité d'Angular:

Génération	Usage
Component	`ng g component my-new-component`
Directive	`ng g directive my-new-directive`
Pipe	`ng g pipe my-new-pipe`
Service	`ng g service my-new-service`
Class	`ng g class my-new-class`
Guard	`ng g guard my-new-guard`
Interface	`ng g interface my-new-interface`
Enum	`ng g enum my-new-enum`
Module	`ng g module my-module`

> Correction du TP : tp1-solution

TP 2 - Le composant Chronomètre

Nous allons dans ce TP créer un composant Chronomètre. Nous verrons ainsi comment créer un composant ainsi que la création d'un Pipe pour formater l'affiche du composant.

Notre premier composant

Dans le dossier `src/app`, créer un dossier `chrono`. Ce dossier contiendra les fichiers `html`, `css` et `ts`.

Voici la sctructure cible du projet :

```
src/app
├── app.component.css
├── app.component.html
├── app.component.spec.ts
├── app.component.ts
├── app.module.ts
└── chrono
```

```
    ├── chrono.component.css
    ├── chrono.component.html
    └── chrono.component.ts
```

Vous devez donc créer les fichiers nécessaires au componsant Chrono !

Ensuite éditez le fichier `chrono.component.ts` et collez le contenu suivant :

```
import { Component, OnInit } from '@angular/core';

@Component({
    selector: 'chrono',
    templateUrl: './chrono.component.html',
    styleUrls: ['./chrono.component.css']
})
export class ChronoComponent implements OnInit {

    constructor() {

    }

    onInit() {

    }
}
```

Puis ajoutez le composant dans la liste des composants de `AppModule` comme suivant :

```
@NgModule({
    declarations: [
        AppComponent,
        ChronoComponent // <- Ici
    ],
    ...
})
export class AppModule { }
```

> Vous devez systématiquement déclarer votre composant en tant que dépendence du module pour l'utiliser !

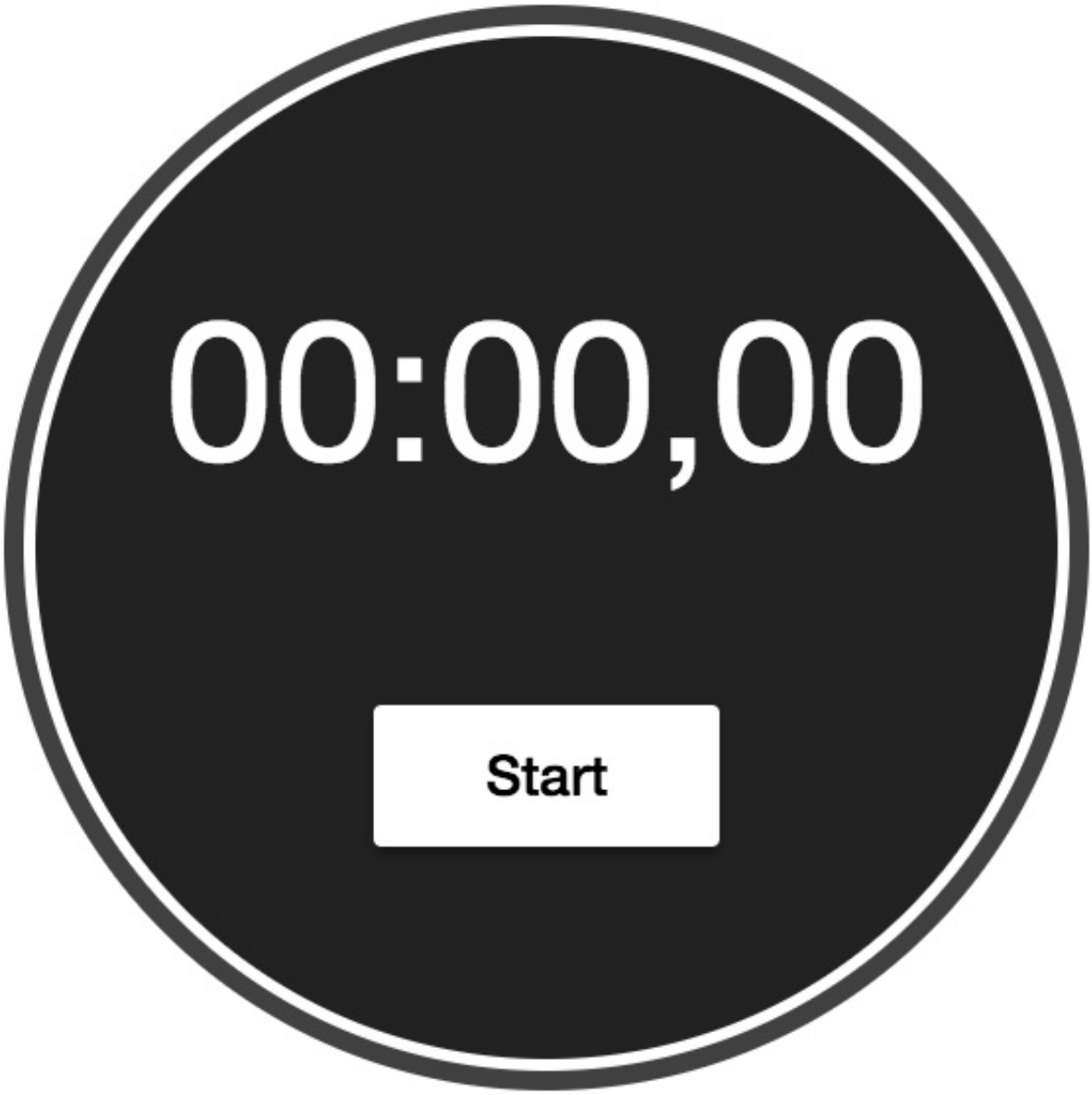
Exercice 1 - Template

Commencez par créer le template HTML nécessaire à l'affichage des données du chronomètre telles que:

- Les millisecondes,
- Les secondes,
- Les minutes.

Et n'oubliez pas le bouton start/stop du chrono !

Voici un exemple de ce qui est attendu :



Vous pouvez utiliser `@angular/material` pour créer un bouton comme suivant :`

```
<button md-raised-button>Text</button>
```

Pour le css :

```
:host {
    text-align: center;
    display: block;
    width: 200px;
    height: 200px;
    margin: auto;
    background: #222;
    padding: 30px;
```

```
    border-radius: 100%;
    position: relative;
    z-index: 0;
}
:host>div {
    position: relative;
    z-index: 3;
}
:host:after {
    content: ' ';
    position: absolute;
    width: 206px;
    height: 206px;
    border: 5px solid #414141;
    border-radius: 100%;
    top: -8px;
    left: -8px;
    padding: 30px;
    z-index: 0;
}
:host .times {
    display: block;
    font-size:50px;
    color:white;
    margin-bottom: 50px;
    padding-top: 30px;
}

:hover button {

}
```

> Note : le selecteur `:host` représente l'élément encapsulant votre composant Angular.

Exercice 2 - Création de l'action click

À partir du cours, essayez de créer une action au click du bouton.

Exercice 3

Maintenant que vous avez créer l'action au click du bouton, implémentez les méthodes `startTimer()` et `stopTimer()`.

Notre premier Pipe

En l'état l'affichage des minutes, secondes et millisecondes n'est pas correct (cf. capture chrono).

Nous allons donc créer un Pipe, équivalent des filtres Angular 1, pour formatter les données.

```
src/app
├── app.component.css
```

```
├── app.component.html
├── app.component.spec.ts
├── app.component.ts
├── app.module.ts
├── chrono
│   ├── chrono.component.css
│   ├── chrono.component.html
│   ├── chrono.component.spec.ts
│   └── chrono.component.ts
└── dec-to-str.pipe.ts
```

Commencez par créer le fichier `dec-to-str.pipe.ts`. Créez la `classe DecToStrPipe` comme présenté dans le cours.

Exercice

Le filtre doit transformer/formatter un entier vers un string sur deux chiffres !

> Correction du TP : tp2-solution

TP 3 - Créer une liste d'utilisateurs en ligne

Dans ce TP, nous allons créer un tableau contenant la liste des utilisateurs avec leur statut en ligne.

Pour rappel, la structure initiale du projet est la suivante :

```
src/app
├── app.component.css
├── app.component.html
├── app.component.spec.ts
├── app.component.ts
├── app.module.ts
```

Notre 1er Service

Dans le dossier `src/app`, créer votre 1er service sous le nom UserService.

Pour aller plus vite, vous pouvez utiliser l'outil `@angular/cli` pour créer votre service.

`ng g service user`

```
src/app
├── user.service.spec.ts
├── user.service.ts
```

Ce service retourne la liste de tous les utilisateurs avec leur statut en ligne :

```
[
    {"id": 1, "email": "john.doe@gmail.com", "status": "online"},
    {"id": 2, "email": "jane.doe@gmail.com", "status": "online"},
    {"id": 3, "email": "jean.dupond@gmail.com", "status": "busy"},
    {"id": 4, "email": "jean.dupont@gmail.com", "status": "offline"},
    {"id": 5, "email": "jeanne.dupond@gmail.com", "status": "offline"},
    {"id": 6, "email": "joe.doe@gmail.com", "status": "online"}
]
```

Compostant tableau des utilisateur

Dans le dossier `src/app` , créer un dossier `user-table` qui contiendra tout le code source du composant UserTableComponent.

Ce composant UserTableComponent affiche la liste des utilisateurs avec leur statut dans tableau HTML.

Pour aller plus vite, vous pouvez utiliser l'outil `@angular/cli` pour créer votre composant.

`ng g component user-table`

Ajouter les feuilles de style à votre composant pour que :

```
    - les lignes pairs soient en fonds gris clair
    - le texte 'online' soit en vert
    - le texte 'offline' soit en gris clair
    - le texte 'busy' soit en rouge
    - l'entête du tableau soit en fonds bleu clair, avec le texte en blanc
```

Cliquer sur le texte descriptif (balise caption) du tableau pour cacher/afficher les utilisateurs hors ligne.

```
    Le texte du caption est sousligné.
    Chaque clic fait permuter les textes suivants :
    "Cliquez sur ce texte pour cacher les utilisateurs hors ligne", texte stylisé en gris clair
    "Cliquez sur ce texte pour afficher les utilsateurs en ligne", texte stylisé en bleu foncé
```

Ci-dessous le code source ajouté au projet :

```
src/app
└── user-table
```

```
    ├── user-table.component.css
    ├── user-table.component.html
    ├── user-table.component.spec.ts
    └── user-table.component.ts
```

Composant enfant

Créer un composant UserTableRowComponent représentant une ligne du tableau.

Pour aller plus vite, vous pouvez utiliser l'outil `@angular/cli` pour créer votre composant.

```
ng g component user-table-row
```

Modifier votre code pour intégrer ce composant dans le composant UserTableComponent.

Ci-dessous le code source ajouté au projet :

```
src/app
└── user-table-row
    ├── user-table-row.component.css
    ├── user-table-row.component.html
    ├── user-table-row.component.spec.ts
    └── user-table-row.component.ts
```

Conclusion

Ce TP vous aura appris à :

- utiliser les directives intégrées à Angular (*ngIf,* ngFor)
- créer un service Angular et l'injecter dans un composant
- créer des composants avec une relation parent-enfant
- créer et utiliser un template