

valtech_

Labs Angular

Sommaire

Nous verrons au cours des travaux pratiques les points suivants :

- [TP 1 - Installation](#)
- [TP 2 - Le composant Chronomètre](#)
- [TP 3 - Créer une liste d'utilisateurs en ligne](#)
- [TP 4 - Formulaires et routes](#)
- [TP 5 - Intégration des webservice](#)
- [TP 6 - Programmation reactive et websocket](#)

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 dépendance 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	<code>ng g component my-new-component</code>
Directive	<code>ng g directive my-new-directive</code>
Pipe	<code>ng g pipe my-new-pipe</code>
Service	<code>ng g service my-new-service</code>
Class	<code>ng g class my-new-class</code>
Guard	<code>ng g guard my-new-guard</code>
Interface	<code>ng g interface my-new-interface</code>
Enum	<code>ng g enum my-new-enum</code>
Module	<code>ng g module my-module</code>

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 composant 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() {

  }

  ngOnInit() {

  }

}
```

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 dépendance 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 créé 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 formater 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
```

Genèrera les fichiers suivants :

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

Ce service devra retourner la liste de tous les utilisateurs :

```
[
  {"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"}
]
```

Le composant tableau des utilisateurs

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
```

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
```

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.

Le tableau possède une checkbox (`md-checkbox`) permettant d'afficher soit les utilisateurs en ligne, soit tous les utilisateurs.

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
```







Le `UserTableRowComponent` devra gérer l'affichage des données de l'utilisateur ainsi que l'action de s'afficher ou se masquer en fonction du statut d'affichage demandé par le composant parent.

Communication entres composants

Notre tableau affiche désormais les données. Nous allons maintenant utiliser l'annotation `@Output` pour émettre un événement lorsque l'utilisateur clique sur le bouton correspondant à la ligne de l'utilisateur. Cet événement sera remonté vers le composant `UserTableComponent` qui remontera cet événement vers le composant `AppComponent`.

Une fois l'information de l'utilisateur remonté dans `AppComponent`, vous devrez afficher les informations de l'utilisateur dans une boîte de dialogue (soit via une alert ou en utilisant la librairie [Material design](#)).

☐ Online only

ID	Email	Status	
1	john.doe@gmail.com	online	
2	jane.doe@gmail.com	online	
3	jean.dupond@gmail.com	busy	
4	jean.dupont@gmail.com	offline	
5	jeanne.dupond@gmail.com	offline	
6	joe.doe@gmail.com	online	

Dialog
john doe is online
Fermer

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,
- Communiquer entre les composants.

Correction du TP : [tp3-solution](#)

TP 4

| Formulaires et routes

Dans ce TP, nous allons créer les formulaires de connexion et d'authentification.

| Correction du TP : [tp4-solution](#)

TP 5

| Intégration des webservices

Dans ce TP, nous allons créer les formulaires de connexion et d'authentification.

| Correction du TP : [tp5-solution](#)

TP 6

| Programmation réactive et websocket

Dans ce TP, nous allons créer les formulaires de connexion et d'authentification.

| Correction du TP : [tp6-solution](#)