



Angular 2 – Forms

1- Objectif :

Dans ce premier Lab, on va découvrir les formulaires avec l'Angular 2.



2- La création d'une nouvelle application :

L'environnement est maintenant en place. On va créer notre application :

- D'abord, on commence par le composant principal « app.component.ts » :

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

@Component({
  selector: 'form-lab',
  templateUrl: "../app/html/Accueil.html"
})

export class AppComponent { }
```

Et le vue du composant « Accueil.html » sous le répertoire « ./app/html »

```
<div class="container" style="margin-top:40px">
<div class="row" style="">
<h3 class="col-lg-12 text-center">
    Application Angular 2 - Lab 1 - Formulaire</h3>
</div>
</div>
```

- Puis, on crée un module racine « app.module.ts »

```
import { NgModule }      from '@angular/core';
import { BrowserModule }  from '@angular/platform-browser';

import { AppComponent }   from './app.component';

@NgModule({
  imports: [BrowserModule],
  declarations: [AppComponent],
  bootstrap: [AppComponent],
  providers: []
})
export class AppModule { }
```

- Ensuite, on va créer un module chargé de démarrer notre application « main.ts » :

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app.module';

const platform = platformBrowserDynamic();

platform.bootstrapModule(AppModule);
```

- Enfin, on va compiler le code TypeScript et lancer l'application avec la commande suivante :

```
npm start
```

Pour information : on a configuré la commande « start » dans le fichier package.json comme suit :

```
"start": "tsc && concurrently \"npm run tsc:w\" \"npm run lite\" "
```

Au final, on aura cet affichage dans le navigateur

Application Angular 2 - Lab 1 - Formulaire

3- Formulaire :

Dans cette partie, nous allons voir comment créer un formulaire.

1. **La couche présentation - Vue** : vous trouvez ci-dessous le contenu d'un fichier html « Formulaire.html » qu'il faut ajouter sous le répertoire « app/html » (ce fichier est un Template Html & CSS (Bootstrap) qui permet de créer un nouveau compte)

```
<div class="container" style="margin-top:40px;background-color: #428bca;">
  <div class="row" style="">
    <h3 class="col-lg-12 text-center">
      <span> <i class="glyphicon glyphicon-user"></i> </span>
      Formulaire pour créer un compte</h3>
    </div>
  </div>
</div>
<form name="formDemo" style="margin-top:40px;">
  <div class="container">
    <div class="row">
      <div class="col-lg-12 col-md-12">
        <div class="form-group col-lg-2 col-lg-offset-1">
          <label class="control-label">Sexe</label>
          <select id="sexe" name="sexe" class="form-control">
            <option value="H">Homme</option>
            <option value="F">Femme</option>
          </select>
        </div>
        <div class="form-group col-lg-4 col-md-5 col-sm-6 col-xs-12">
          <label class="control-label" for="prenom">Prénom</label>
          <input type="text" class="form-control" id="prenom"
name="prenom">
        </div>
        <div class="form-group col-lg-4 col-md-5 col-sm-6 col-xs-12">
          <label class="control-label" for="nom">Nom</label>
          <input type="text" class="form-control" id="nom" name="nom">
        </div>
      </div>
      <div class="col-lg-12 col-md-12">
        <div class="form-group col-lg-10 col-md-10 col-lg-offset-1">
          <label class="control-label" for="email">Adresse e-
mail</label>
          <input type="text" class="form-control" id="email"
name="email">
        </div>
      </div>
    </div>
    <div>
      <div class="text-center">
        <div class="btn-group">
          <button type="button" id="btnValide" name="btnValide"
class="btn btn-primary center-block"> Valider </button>
          <button type="button" id="btnAnnuler" name="btnAnnuler"
class="btn btn-info center-block"> Annuler </button>
        </div>
      </div>
    </div>
  </div>
</form>
```

2. Le composant du formulaire :

- On va créer une nouvelle classe Client « client.ts » avec le schéma suivant :

```
export class Client {
  public Nom :string;
  public Prenom :string;
  public AdresseMail : string;
  public Sexe :string;

  constructor() {
    this.Sexe="";
    this.Prenom = "";
    this.AdresseMail = "";
    this.Nom = "";
  };
}
```

- On doit déclarer un nouveau composant « client.form.component.ts » qui permet de gérer les clients

```
import { Component } from '@angular/core';
import { Client } from './client';

@Component({
  selector: 'client-form',
  templateUrl: 'app/html/Formulaire.html'
})

export class ClientFormComponent {
  private Sexes = [ { Code: 'H', Libelle: 'Homme' },
                    { Code: 'F', Libelle: 'Femme' } ];

  public client:Client = new Client();

  constructor() {
  }

  onClick_Valider(value:Client): void {
    console.log(value);
  }

  onClick_Annuler(): void {
    console.log("Annuler");
  }
}
```

Maintenant, il faut inclure le nouveau composant dans @NgModule afin de qu'on puisse l'utiliser dans notre application. Ouvrez le fichier « app.module.ts » et ajoutez le code suivant :

```
import { NgModule }           from '@angular/core';
import { BrowserModule }      from '@angular/platform-browser';
import { FormsModule, FormGroup } from '@angular/forms';

import { AppComponent }       from './app.component';
import { ClientFormComponent } from './client.form.component';

@NgModule({
  imports: [BrowserModule, FormsModule, FormGroup],
  declarations: [AppComponent, ClientFormComponent],
  bootstrap: [AppComponent],
  providers: []
})
export class AppModule { }
```

Il faut modifier la page « Accueil.html », pour implémenter le formulaire client :

```
<div class="container" style="margin-top:40px">
  <div class="row" style="">
    <h3 class="col-lg-12 text-center">
      Application Angular 2 - Lab 1 - Formulaire</h3>
    </div>
  </div>

  <client-form></client-form>
```

- On va s'intéresser à l'implantation d'Angular dans notre page « Formulaire.html » :
 - Ajoutez la directive NgModel dans les trois inputs de l'exemple comme ci-dessous :

```
<input type="text" class="form-control" id="prenom" name="prenom"
  [(ngModel)]="client.Prenom">
```

```
<input type="text" class="form-control" id="nom" name="nom"
  [(ngModel)]="client.Nom">
```

```
<input type="text" class="form-control" id="email" name="email"
  [(ngModel)]="client.AdresseMail">
```

- Affichez le nom et le prénom dans le titre de la page:

```
<div class="row">
  <h3 class="col-lg-12 text-center">
    <span> <i class="glyphicon glyphicon-user"></i>
  </span>
    Formulaire pour créer un compte : {{client.Prenom}}
    {{client.Nom}}</h3>
  </div>
```

- Pour l'élément « select », on va ajouter la directive NgModel et *ngFor qui permet de répéter l'élément option tant qu'il y a des entrées dans le tableau Sexes.

```
<select id="sexe" name="sexe" class="form-control"
  [(ngModel)]="client.Sexe">
  <option *ngFor="let s of Sexes"
    [value]="s.Code">{{ s.Libelle }}</option>
</select>
```

- Pour valider le formulaire, on implémente les fonctions permettant de gérer les clients à l'aide de l'événement « click ».

```
<div class='text-center'>
  <div class="btn-group">
    <button type="button" id="btnValide" name="btnValide" class="btn btn-
primary center-block" (click)="onClick_Valider(client)"> Valider </button>
    <button type="button" id="btnAnnuler" name="btnAnnuler" class="btn btn-
info center-block" (click)="onClick_Annuler()"> Annuler </button>
  </div>
</div>
```

On a utilisé les syntaxes suivantes :


- La syntaxe [] permet de synchroniser les données depuis le model vers la view.
- La syntaxe () permet de synchroniser un événement depuis la view vers le model.
- La combinaison des deux [] nous permet de faire un double data binding.

Vous pouvez modifier l'objet client dans le constructeur, comme suit afin de tester l'affichage :

```
this.client.Nom = "CHAABANE";
this.client.Prenom = "Ramy";
this.client.Sexe="H";
this.client.AdresseMail="test@gmail.com";
```

Et voici le résultat sur le navigateur :

Application Angular 2 - Lab 1 - Formulaire

 Formulaire pour créer un compte : Ramy CHAABANE

Sexe	Prénom	Nom
<input type="text" value="Homme"/>	<input type="text" value="Ramy"/>	<input type="text" value="CHAABANE"/>
Adresse e-mail		
<input type="text" value="...@gmail.com"/>		
<input type="button" value="Valider"/> <input type="button" value="Annuler"/>		

4- Validation du formulaire :

On définit les règles suivantes :

- Le nom, Prénom et sexe sont obligatoires.
- Le nom et prénom doit contenir au max 50 caractères et au min 2 caractères.
- L'email doit être sous forme xxx@xxx.xxx

Angular prend en charge les formulaires et leur validation.

Les règles de validation de base supportées sont :

```
<input [type="email | number | month | url | ..."]  
      [required=""]  
      [minlength=""]  
      [maxlength=""]  
      [pattern=""]>  
</input>
```

Pour cela plusieurs propriétés sont accessibles au niveau du formulaire et des champs qui le compose :

- **valid** : Indique si les règles de validation sont respectées pour l'élément actuel
- **invalid** : Indique si au moins une des règles de validation est violée pour l'élément actuel
- **pristine** : Indique si l'élément actuel n'a pas été modifié
- **dirty** : Indique si l'élément actuel a été modifié
- **touched** : Indique si l'élément actuel a été touché.

Ces propriétés sont accessibles via :

- <nom du formulaire>. <propriété>
- <nom du champ>. <propriété>
- <nom du champ>.errors. <propriété d'erreur>

Ajouter les directives de validation au Template :

```
<form name="formDemo" style="margin-top:40px;">
  <div class="container">
    <div class="row">
      <div class="col-lg-12 col-md-12">
        <div class="form-group col-lg-2 col-lg-offset-1">
          <label class="control-label">Sexe</label>
          <select id="sexe" name="sexe" class="form-control" required
            [(ngModel)]="client.Sexe" >
            <option *ngFor="let s of Sexes" [value]="s.Code">{{ s.Libelle=}} </option>
          </select>
        </div>
        <div class="form-group col-lg-4 col-md-5 col-sm-6 col-xs-12">
          <label class="control-label" for="prenom">Prénom</label>
          <input type="text" class="form-control" id="prenom" name="prenom"
            required maxlength="50" minlength="2" [(ngModel)]="client.Prenom">
        </div>
        <div class="form-group col-lg-4 col-md-5 col-sm-6 col-xs-12">
          <label class="control-label" >Nom</label>
          <input type="text" class="form-control" id="nom" name="nom"
            required maxlength="50" minlength="2" [(ngModel)]="client.Nom" >
        </div>
      </div>
      <div class="col-lg-12 col-md-12">
        <div class="form-group col-lg-10 col-md-10 col-lg-offset-1">
          <label class="control-label" for="email">Adresse e-mail</label>
          <input type="email" class="form-control" id="email" name="email" required
            pattern="^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$"
            [(ngModel)]="client.AdresseMail">
        </div>
      </div>
    </div>
    <div>
      <div class="text-center">
        <div class="btn-group">
          <button type="button" id="btnValide" name="btnValide"
            class="btn btn-primary center-block"
            (click)="onClick_Valider('Client')" > Valider </button>
          <button type="button" id="btnAnnuler" name="btnAnnuler"
            class="btn btn-info center-block"
            (click)="onClick_Annuler()" > Annuler </button>
        </div>
      </div>
    </div>
  </div>
</form>
```

Maintenant toutes les règles de validation sont en place, pour afficher un message d'erreur à l'utilisateur lorsqu'une règle de validation d'un champ n'est respectée, ci-dessous un exemple pour le champ Prénom.

Il y a une syntaxe particulière très utilisée pour déclarer une variable dans le code HTML : il suffit de précéder le nom de la variable par un#

```
<div class="form-group col-lg-4 col-md-5 col-sm-6 col-xs-12">
  <label class="control-label" for="prenom">Prénom</label>
  <input type="text" class="form-control" id="prenom" name="prenom" required
  maxlength="50" minlength="6" [(ngModel)]="client.Prenom" #prenom>
</div>
<pre> Class CSS : <br/> {{prenom.className}} </pre>
<pre> Valeur :    {{prenom.value }}</pre>
```

La façon la plus facile de comprendre comment Angular 2 change l'état de l'input : on va visualiser par exemple les classes CSS du champ « Prénom » :

Prénom

Class CSS :
form-control ng-untouched ng-pristine ng-invalid

Valeur :

Par défaut, on a :

form-control ng-untouched ng-pristine ng-invalid

Si vous cliquez à l'intérieur puis à l'extérieur de l'input, vous constatez que l'Angular a supprimé la classe « ng-untouched » et a ajouté la classe « ng-touched »

form-control ng-pristine ng-invalid ng-touched

Si on saisit une valeur valide :

form-control ng-touched ng-dirty ng-valid

On pourra aussi savoir si la valeur a été modifiée (**ng-dirty** / **ng-pristine**) et si le contrôle a été visité (**ng-touched** / **ng-untouched**).


Comme nouveautés par rapport à ce que nous avons vu jusqu'à présent on a du style ajouté en plus dans notre composant « client.form.component » :

```
@Component ({
  selector: 'client-form',
  templateUrl: 'app/html/Formulaire.html',
  styles: [ `
    .ng-valid { border-color: green; }
    .ng-invalid { border-color: red; }
  ` ]
})
```

On a défini deux classes :

- **ng-valid** : si on a une valeur valide, on va ajouter une bordure verte à l'input.
- **ng-invalid** : si on a une valeur non valide, on va ajouter une bordure rouge à l'input.

Voici le résultat, la bordure de l'input change selon à l'état du champ :

 Formulaire pour créer un compte : Chaabane Ramy

Sexe

Prénom

Nom

Adresse e-mail

Valider

Annuler

Maintenant, il faut afficher un texte d'avertissement qui contient un message détaillé sur l'erreur de validation. Pour cela on va :

- Référencer la directive ngModel à notre variable #prenom :

```
<div class="form-group col-lg-4 col-md-5 col-sm-6 col-xs-12">
  <label class="control-label" for="prenom">Prénom</label>
  <input type="text" class="form-control" id="prenom" name="prenom" required
  maxlength="50" minlength="6" [(ngModel)]="client.Prenom" #prenom="ngModel">
</div>
<pre *ngIf="prenom.errors">{{ prenom.errors | json }}</pre>
```

Prénom

```
{
  "required": true
}
```

Prénom

```
{
  "minlength": {
    "requiredLength": 2,
    "actualLength": 1
  }
}
```

- Ajouter les messages d'erreur dans la page HTML « Formulaire.html » :

```
<div class="form-group col-lg-4 col-md-5 col-sm-6 col-xs-12"
  [ngClass]="{'has-error': !prenom.valid, 'has-success': prenom.valid}">
  <label class="control-label" for="prenom">Prénom</label>
  <input type="text" class="form-control" id="prenom" name="prenom"
    required maxlength="50"
    minlength="2" [(ngModel)]="client.Prenom" #prenom="ngModel">

  <div *ngIf="prenom.errors && prenom.touched">
    <span class="text-danger" *ngIf="prenom.errors.minlength">
      Le prénom doit être au minimum 2 caractères </span>
    <span class="text-danger" *ngIf="prenom.errors.required">
      Le prénom est obligatoire </span>
  </div>
```

Nous venons de créer notre premier formulaire en Angular 2 :

 Formulaire pour créer un compte : chaabane

Sexe

Le sexe est obligatoire

Prénom

Le prénom est obligatoire

Nom

Adresse e-mail

L'email n'est pas valide

Valider

Annuler

Formulaire pour créer un compte : rami chaabane

Sexe	Prénom	Nom
<input type="text" value="Homme"/>	<input type="text" value="rami"/>	<input type="text" value="chaabane"/>
Adresse e-mail		
<input type="text" value="test@gmail.com"/>		
<input type="button" value="Valider"/> <input type="button" value="Annuler"/>		



On va désactiver le bouton « Valider » s'il y a une erreur de validation dans le formulaire :

- Ajouter une variable local de type « NgForm » :

```
<form name="formDemo" style="margin-top:40px;" #formClient="ngForm">
```

- Désactiver le bouton « Valider » :

```
<div>
  <div class='text-center'>
    <div class="btn-group">
      <button type="button" id="btnValide" name="btnValide" class="btn btn-
primary center-block" (click)="onClick_Valider(formClient)"
[disabled]="formClient.invalid"> Valider </button>
      <button type="button" id="btnAnnuler" name="btnAnnuler" class="btn btn-info
center-block" (click)="onClick_Annuler()"> Annuler </button>
    </div>
  </div>
```

- Réinitialiser le formulaire après la validation :

```
import { NgForm } from '@angular/forms';

...
onClick_Valider(form:NgForm): void {
  console.log(form.value);
  form.resetForm();
}
```