Les formulaires

Rappel HTML

Un formulaire est déclaré dans le code html grâce à la balise <form>

Il est généralement composé d'un ou plusieurs champs ainsi que d'un bouton pour soumettre le formulaire. L'attribut name est obligatoire pour un champ dans un formulaire

Exemple:

```
<form>
    Nom : <input type="text" name="nom">
    Prenom: <input type="text" name="prenom">
  <button type="submit">Submit</button>
</form>
```

Contraintes sur les champs

Il est possible d'indiquer dans notre HTML des contraintes à mettre sur les champs, certaines contraintes sont des standards HTML:

- Marquer un champ comme obligatoire
- Définir la longueur maximale du texte attendu dans le champ
- Définir la longueur minimale du texte attendu dans le champ
- Définir le format attendu dans le champ

Angular vous permet même d'implémenter vos propres contraintes (par exemple on pourrait vérifier qu'un nom d'utilisateur est présent dans notre base de données lors d'une inscription)

Exemple d'utilisation des contraintes

```
<form>

        Nom : <input type="text" name="nom" required maxlength="15" minlength="2">

        Prenom: <input type="text" name="prenom" pattern="[a-zA-Z]+">

      <button type="submit">Submit</button>

    </form>
```

On voit sur l'exemple que le champ "nom" est requis et doit etre compris entre 2 et 15 caractères

Le champ "prenom" ne doit contenir que des lettres

Formulaires

Les formulaires sont au coeur de chaque applications : utilisés pour s'enregistrer, se connecter, réserver son billet d'avion, planifier une réunion...

Angular permet de gérer les formulaires de 2 manières différentes:

- "**Template Driven Form**" (ce que nous allons voir en cours) - utilisé pour les formulaires simple, il faut importer le **FormsModule** dans notre application pour l'utiliser
- "**Reactive form**" - Pour les formulaires plus complexes. Il faut importer **ReactiveFormsModule** pour pouvoir l'utiliser.

Avec Angular

Quand Angular croise un formulaire dans votre HTML, il va automatiquement créer un objet de type **NgForm** pour le formulaire (https://angular.io/api/forms/NgForm)

Il va aussi créer un objet de type **FormControl**(https://angular.io/api/forms/FormControl) pour chacune de vos balises <input> sur le formulaire.

Ces objets vont nous permettre de savoir si notre formulaire est valide suivant les contraintes qu'on aura mis en place (tel champ est requis, tel champ à une longueur maximale de 5 caracteres..)

Cela va nous permettre d'afficher un message d'erreur à l'utilisateur le cas échant.

Propriétés interessantes

NgForm et FormControl ont des propriétés intéressantes que nous pouvons exploiter dans notre HTML:

- dirty: boolean -> true si l'utilisateur a changé la valeur du champ/formulaire
- touched: boolean -> true si l'utilisateur a touché le champ/formulaire
- errors : ValidationErrors | null -> objet contenant les erreurs si il y en a
- valid: boolean -> true si le champ/formulaire respecte ses contraintes
- invalid: boolean -> l'inverse de valid

Avec Angular

On peut récupérer ces objets dans notre template, grâce aux variables de template (#)

```
<form #varForm="ngForm">

        Nom : <input type="text" required maxlength="24" name="nom" [(ngModel)]="nom"
#nomVariable="ngModel">

<div *ngIf="nomVariable.dirty && nomVariable.invalid">

    <div *ngIf="nomVariable.errors.required">Ce champ est requis</div>

</div>

        Prenom: <input type="text" name="prenom" [(ngModel)]="prenom"
#prenomVariable="ngModel">

      <button type="submit" [disabled]="varForm.invalid">Submit</button>

    </form>
```

Soumission du formulaire

Lors de la soumission du formulaire, la balise <form> emet un evenement "ngSubmit" auquel on doit réagir pour gérer la soumission.

Exemple:

```
<form (ngSubmit)="onSubmit()" #heroForm="ngForm">

...

</form>
```

Avec style

Angular rajoute automatiquement des classes CSS à vos inputs en fonction de leur état. Si vous les définissez vous pouvez donc personnaliser l'affichage de vos champs suivant leur état:

ng-valid -> classe appliquée si le champ est valide

ng-invalid -> classe appliquée si le champ est invalide

ng-dirty -> classe appliquée si la valeur du champ a été modifée

..

Le reste est ici : https://angular.io/guide/form-validation#control-status-css-classes