Chapitre 14
Gestion avancée des formulaires

1. Commandes Git

Les commandes GIT suivantes permettent d'accéder au code source des chapitres précédents :

```
cd C:\
git clone https://github.com/EditionsENI/JavaScriptEtAngular.git
cd C:\JavaScriptEtAngular\Angular
git checkout chapitre14
code .
```

2. Introduction

La création d'un formulaire est une étape cruciale, mais maîtrisée, du développement d'une application web. La validation, la mise en forme et l'interaction avec l'utilisateur sont des phases beaucoup plus chronophages et problématiques que ne le laisse supposer le chapitre Formulaires, à tel point qu'elles en méritaient bien un tout entier.

3. Validation des champs de saisie

Le cadriciel laisse l'opportunité à un développeur de créer des validateurs de champs de saisie répondant à ses besoins métier. Angular en propose quelques-uns afin de répondre à la plupart des problématiques des applications web modernes. Chacun de ces validateurs est une méthode provenant de la classe `Validators`.

Pour un champ de n'importe quel type :

– `required`() permet d'indiquer que le champ est requis.

– `pattern`(`regEx`) permet de valider l'entrée saisie uniquement si elle correspond à l'expression régulière passée en paramètre.

Pour un champ de type texte :

– `minLength`(`nombre`) permet de s'assurer que la chaîne respecte le nombre minimum de caractères ;

– `maxLength`(`nombre`) permet de s'assurer que la chaîne n'a pas plus qu'un certain nombre de caractères.

Pour une adresse e-mail :

– `email`() permet de s'assurer que la chaîne entrée est bien une adresse e-mail.

Pour un nombre :

– `min`(`nombre`) permet de définir une valeur minimum ;

– `max`(`nombre`) permet de définir une valeur maximum.

3.1 Dans un formulaire basé sur un template

Lorsque c'est la première méthode de création d'un formulaire qui a été choisie, la mise en place des validateurs se fait également dans le gabarit en rajoutant des attributs à l'élément HTML concerné.

```
<elementHTML ngModel unValidateur unAutre></elementHTML>
ng generate component composants/C14
ng generate class modeles/monFormulaire
```

Puis :

```
// C:\JavaScriptEtAngular\Angular\monProjetAngular\src\app\
composants\c14\c14.components.ts
import { Component, OnInit } from '@angular/core';
import { MonFormulaire } from '../../modeles/mon-formulaire';

@Component({
 selector: 'app-c14',
 template: `
 <div class="container text-center">
     <h2>Le composant C14</h2>
     <hr/>
     <div class="row justify-content-md-center">
       <div>
            <form #nomDuFormulaire="ngForm">
                <div>Valeurs du modèle : {{formulaire | json}}
</div>
                <hr/>
                <div>Formulaire valide ? : {{nomDuFormulaire.
valid}}</div>
                <hr/>
                <div class="form-group">
                    <label>Nom : </label>
                    <input type="text" class="form-control"
name="nom" [(ngModel)]="formulaire.nom" required minlength="2"
                           maxlength="25">
                </div>
                <div class="form-group">
                  <label>Couleur : </label>
                  <select class="form-control" name="couleur"
[(ngModel)]="formulaire.couleur">
                  <option selected>Blanc</option>
                  <option>Bleu</option>
                  <option>Noir</option>
                  <option>Rouge</option>
                  <option>Vert</option>
                  </select>
                </div>
                <div class="form-check">
                  <input class="form-check-input" type="checkbox"
name="majeur" [(ngModel)]="formulaire.majeur">
                  <label class="form-check-label">Je suis majeur
et vacciné </label>
                </div>
                <button type="submit">Envoyer</button>
```

```
            </form>
            <hr/>
        </div>
`

})
export class C14Component implements OnInit {

 formulaire = new MonFormulaire('', '', false);

 constructor() { }

 ngOnInit() {
 }

}
```

Interpoler la valeur de `nomDuformulaire.valid` permet de savoir si le formulaire est valide, ou non. Le fait qu'un formulaire ne soit pas valide n'empêche pas un utilisateur de le soumettre. Pour effectivement bloquer la soumission d'un formulaire, il faudrait, par exemple, désactiver le bouton de soumission tant que le formulaire n'est pas valide.

```
<button type="submit" [disabled]=" nomDuFormulaire.invalid">
Envoyer</button>
```

Utilisation d'un validateur Angular dans un formulaire basé sur un gabarit

3.2 Dans un formulaire basé sur une classe TypeScript

Lorsque c'est la seconde méthode qui a été choisie, la mise en place des validateurs se fait dans la classe TypeScript. Les mêmes validateurs vont être utilisés, mais d'une façon différente. À noter que les validations dans le gabarit sont possibles même si les champs de formulaire sont gérés en TypeScript. L'inverse n'est pas vrai.

```
attribut: new FormControl('', validateur)
```

Si plusieurs validateurs doivent s'appliquer, il est obligatoire de les mettre dans un tableau :

```
attribut: new FormControl('', [validateur1, validateur2,
validateur3])
```

```
// C:\JavaScriptEtAngular\Angular\monProjetAngular\src\app\
composants\c14\c14.components.ts
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from
'@angular/forms';

@Component({
 selector: 'app-c14',
 template: `
 <div class="container text-center">
 <h2>Le composant C14</h2>
 <hr/>
 <div>Formulaire valide ? : {{monFormulaire.valid}}</div>
 <hr/>
 <div class="row justify-content-md-center">
     <div>
         <form [formGroup]="monFormulaire">
             <div class="form-group">
                 <label>Nom : </label>
               <input type="text" class="form-control" name="nom"
formControlName="nom">
             </div>
             <div class="form-group">
                 <label>Couleur : </label>
                 <select class="form-control" name="couleur"
formControlName="couleur">
                     <option selected>Blanc</option>
```

```
                    <option>Bleu</option>
                    <option>Noir</option>
                    <option>Rouge</option>
                    <option>Vert</option>
                </select>
            </div>
            <div class="form-check">
                <input class="form-check-input" type="checkbox"
name="majeur"formControlName="majeur">
                <label class="form-check-label">Je suis majeur
et vacciné
</label>
            </div>
            <button type="submit">Envoyer</button>
        </form>
        <hr/>
    </div>
 </div>
</div>
`
})
export class C14Component implements OnInit {
  monFormulaire = new FormGroup({
   nom: new FormControl('', [Validators.required,
Validators.maxLength(25)]),
   couleur: new FormControl('', Validators.required),
   majeur: new FormControl('', Validators.required)
 });


 constructor() { }

 ngOnInit() {
 }

}
```

4. Validateurs personnalisés

Parfois, les validateurs Angular ne suffisent pas à retransmettre parfaitement les spécificités de l'application. Soit parce qu'elles ont été complexifiées volontairement par le chef de projet, soit parce que l'environnement métier est particulièrement délicat.

Fort heureusement, le développeur peut créer autant de validateurs personnalisés qu'il le souhaite. Un validateur est une méthode TypeScript qui prend en paramètre une instance de `FormControl` et qui retourne une erreur, ou rien.

```
monValidateur(champ: FormControl) {
   if (condition) {
     return {erreur: true};
   } else {
     return null;
   }
 }
```

```
// C:\JavaScriptEtAngular\Angular\monProjetAngular\src\app\
composants\c14\c14.components.ts
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from
'@angular/forms';

@Component({
 selector: 'app-c14',
 template: `
 <div class="container text-center">
 <h2>Le composant C14</h2>
 <hr/>
 <div>Formulaire valide ? : {{monFormulaire.valid}}</div>
 <hr/>
 <div class="row justify-content-md-center">
     <div>
         <form [formGroup]="monFormulaire">
             <div class="form-group">
                 <label>Nom : </label>
                 <input type="text" class="form-control" name="nom"
formControlName="leNom">
             </div>
             <div class="form-group">
                 <label>Couleur : </label>
                 <select class="form-control" name="couleur"
formControlName="laCouleur">
                     <option selected>Blanc</option>
                     <option>Bleu</option>
                     <option>Noir</option>
                     <option>Rouge</option>
                     <option>Vert</option>
```