[CGA-JAVA-WFDA] Web Front-end Development Angular 2.1

[Bài đọc] Template Driven Form

Các loại form trong Angular

Có 2 loại Form trong Angular là Template-driven form và Model Driven form

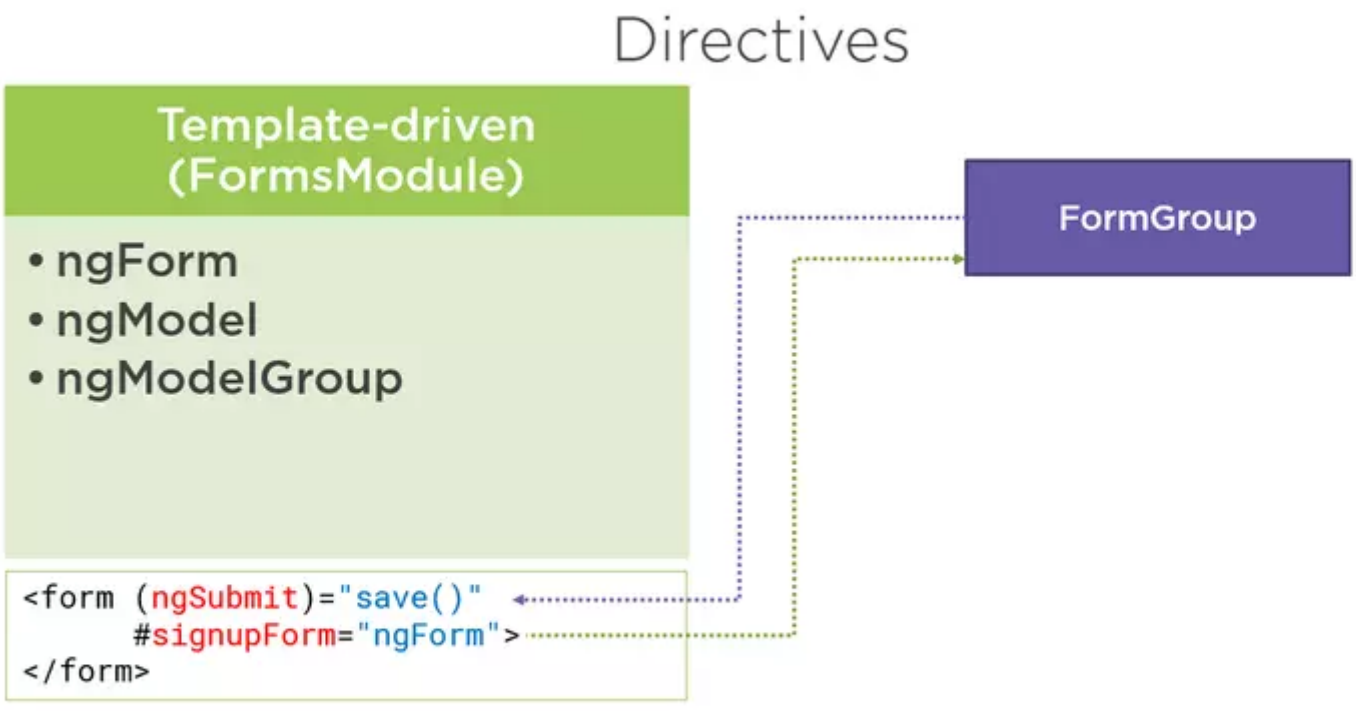
- Template-driven form là tạo ra các form dựa vào template bằng cách thực hiện việc thêm các directives và các hành vi vào template, sau đó Angular sẽ tự động tạo forms để quản lý và sử dụng.
- Model Driven Form hay còn gọi là Reactive Form, phương pháp này tránh việc sử dụng các directive ví dụ như ngModel, required,..., thay vào đó tạo các Object Model ở trong các Component, rồi tạo ra form từ chúng.

Template-driven forms

Template-Driven Form tạo form dựa vào template, trong trường hợp này thì Form Model có thể nói là toàn bộ HTML Form template (toàn bộ đoạn code trong thẻ <form>)

Các directives mà Angular cung cấp: ngForm, ngModel, ngModelGroup...

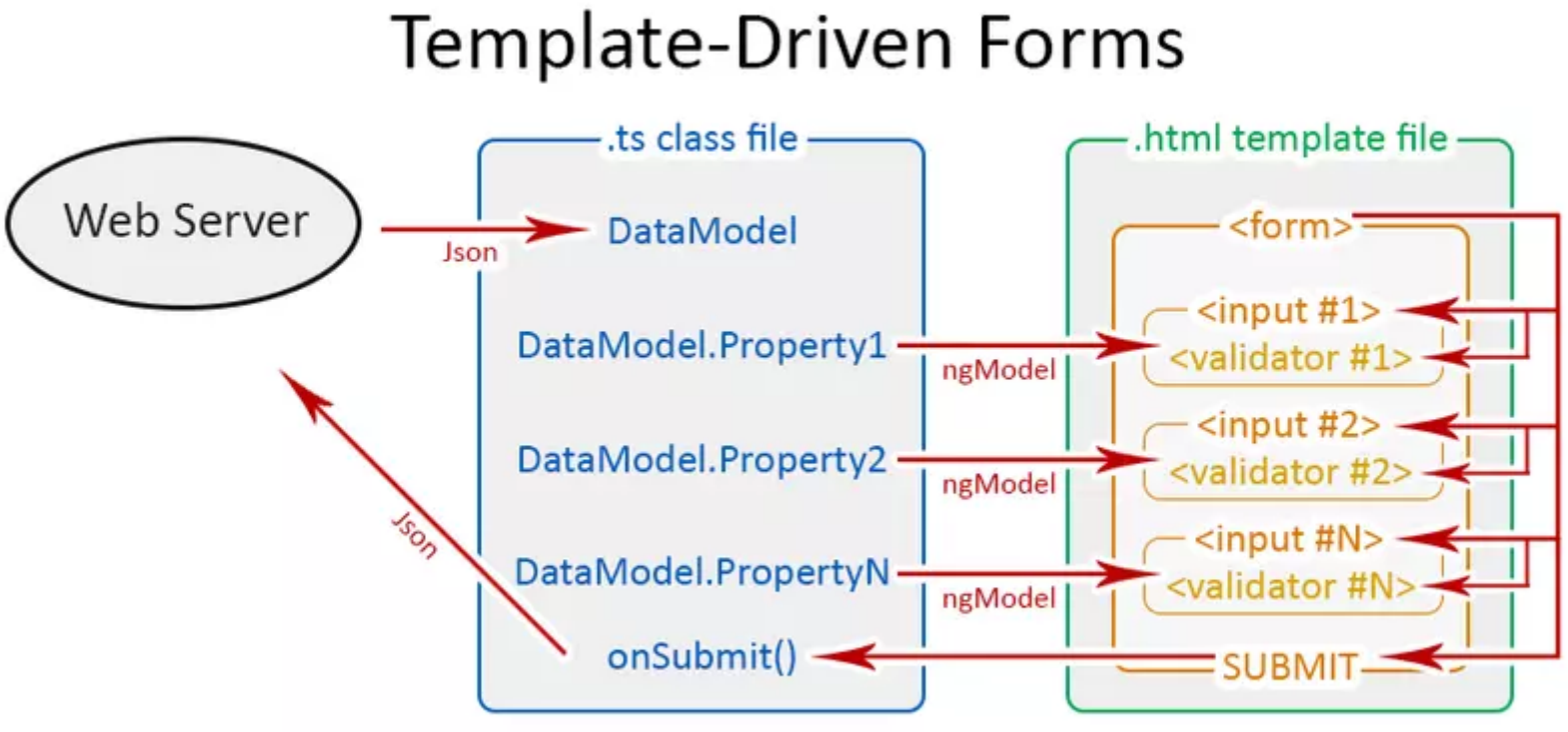
Cấu trúc cơ bản của 1 Template-Driven Form:



Luồng hoạt động

1. User thay đổi dữ liệu trong các ô input như là input, checkbox, text area...
2. Input Element sẽ gửi đi input event chứa value từ user vừa nhập vào
3. Control_Value_Accessor sẽ trigger phương thức setValue( ) trên FormControl instance.
4. FormControl instance sẽ emit giá trị mới qua valueChanges observable
5. Bất cứ subcriber đến valueChanges observable đều nhận được giá trị mới nhất của control đó.
6. Control_Value_Accessor cũng gọi phương thức NgModel.viewToModelUpdate( ) để emit ngModelChange event.
7. Ttrong trường hợp sử dụng cơ chế two-way binding thì lúc này property trong component sẽ được update value bởi ngModelChange event).
8. Khi property trong Component thay đổi thì cơ chế phát hiện thay đổi bắt đầu chạy và control_Value_Accessor sẽ update Input Element trong view với giá trị mới nhất của property trong component.

Flow tổng quát của Template-Driven Form:



Luồng hoạt động

Ví dụ: Để có thể sử dụng các APIs mà Angular cung cấp cho việc thao tác với Template-driven forms, cần phải import NgModule là FormsModule từ package @angular/forms như sau:

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

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

Đầu tiên truy cập vào form instance bằng cách sử dụng ngForm như sau:

```
<form
  autocomplete="off"
  #bookForm="ngForm"
>
</form>
```

Ở trong đoạn code phía trên, chúng ta đã tạo một template variable là bookForm, nó sẽ là một instance của directive ngForm, như thế chúng ta có thể sử dụng các public API mà directive này cung cấp như lấy ra value của nó chẳng hạn như bookForm.value hay là kiểm tra xem form có valid hay không bookForm.invalid

Tuy nhiên thì hiện tại form trên vẫn đang chỉ là 1 form rỗng, công việc tiếp theo là chúng ta phải nói cho Angular biết các form control nào cần phải quản lý. Đây chính là lúc dùng đến ngModel directive.

Chúng ta sẽ thêm ngModel vào các control như sau:

```
<form
  autocomplete="off"
  #bookForm="ngForm"
>
  <div class="form-group">
    <label for="bookName">Book Name</label>
    <input name="name" ngModel type="text" class="form-control">

    <label for="year">Year</label>
    <input  name="year" ngModel type="text" class="form-control">

    <label for="stars">Stars</label>
    <input  name="stars" ngModel type="text" class="form-control">
  </div>
</form>
```

Chú ý là chúng ta cần phải khai báo attribute name cho các form control nhé, bởi vì form sẽ quản lý các form control dựa trên attribute name. Chúng ta thử sử dụng bookForm.value để xem bookForm có những value nào nhé:

```
<pre>{{ bookForm.value | json }}</pre>
```

Kết quả:

```
{
  "name": "",
  "year": "",
  "stars": ""
}
```

Giả sử trong component có sẵn 1 biến book, và chúng ta muốn bind data cho các control với book, lúc này chúng ta sẽ dùng đến binding cho property, và property chúng ta nhắc đến ở đây chính là ngModel.

```
// *.component.ts
book = {name: '123', year: 1993, stars: null};

// *.component.html
...
<input name="name" [ngModel]="book.name" type="text" class="form-control">
...
<input name="year" [ngModel]="book.year" type="text" class="form-control">
...
<input name="stars" [ngModel]="book.stars" type="text" class="form-control">
...
```

Chú ý rằng, khi update form control, bản thân control được form quản lý sẽ thay đổi – bookForm.value, nhưng object book ở trong component sẽ không có thay đổi gì cả, vì chúng ta không hề đụng chạm gì tới nó, chúng ta chỉ binding một chiều, để có thể thay đổi book trong component chúng ta buộc phải binding ngược trở lại. Cách binding trên còn được biết đến với tên gọi two-way binding [(ngModel)].

```
...
<input name="name" [(ngModel)]="book.name" type="text" class="form-control">
...
<input name="year" [(ngModel)]="book.year" type="text" class="form-control">
...
<input name="stars" [(ngModel)]="book.stars" type="text" class="form-control">
...
```

Để thêm các validation cho form thì trong Angular có cung cấp một số Validators cơ bản mà chúng ta có thể dùng ngay trong template như: required, minlength, maxlength... Chúng được viết là các directives, nên chúng ta có thể sử dụng như các directives khác trong template của bạn.

Ví dụ:

```
<input name="name" required minlength="3" [(ngModel)]="book.name" type="text" class="form-control">
```

Chúng ta vừa thêm 2 validators cho form control name đó là form này bắt buộc phải required và có ít nhất là 3 ký tự. Để kiểm tra và dễ dàng quan sát, chúng ta sẽ thêm phần hiển thị lỗi như sau:

```
<pre>{{ bookForm.controls.name?.errors | json }}</pre>
```

Khi ô input này để trống thì ta có thể thấy lỗi hiển thị trên màn hình:

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

hoặc khi nhập vào 2 ký tự, thì kết quả sẽ là:

```
{
  "minlength": {
    "requiredLength": 3,
    "actualLength": 2
  }
}
```

Khi input này được nhập nhiều hơn 2 ký tự thì chúng ta sẽ thấy key required và minlength của object trên sẽ bị xóa bỏ. Chúng ta có thể sử dụng điều này để điều khiển giao diện ẩn/hiển thị lỗi ra view cho người dùng.

Cuối cùng để hoàn thiện form trên thì chúng ta sẽ thêm 1 button dùng để submit form bằng cách sử dụng directive ngSubmit

```
<form
  autocomplete="off"
  #bookForm="ngForm"
  (ngSubmit)="addBook(bookForm.value)"
>
  <div class="form-group">
      ...
  </div>
  <button type="submit" class="btn btn-primary mr-1">Save</button>
</form>
```

Last modified: Saturday, 6 March 2021, 1:14 PM

Previous Activity

Jump to...

Next Activity