9강. form

DEVELOP ACROSS ALL PLATFORMS

Learn one way to build applications with Angular and reuse your code and abilities to build apps for any deployment target. For web, mobile web, native mobile and native desktop.

09 강.

폼(form)

❑ 1) 폼(form) 개요

* 폼 (form)

-폼(form)이란 서버에 데이터를 전달하기 위한 사용자 입력 영역을 의미.
-폼을 이용하면 위젯을 읽고 쓰며 데이터의 유효성을 체크하는 기능이 가능.

* 폼 검증 방식 2가지

가. template 기반

HTML5에서 제공하는 속성 이용한 방식. (required, pattern 등)

나. model 기반

컴포넌트 클래스 내에서 선언한 모델 이용한 방식.

https://angular.io/guide/forms-overview

❑ 2) template 기반 폼 검증

1) FormsModule 모듈 등록

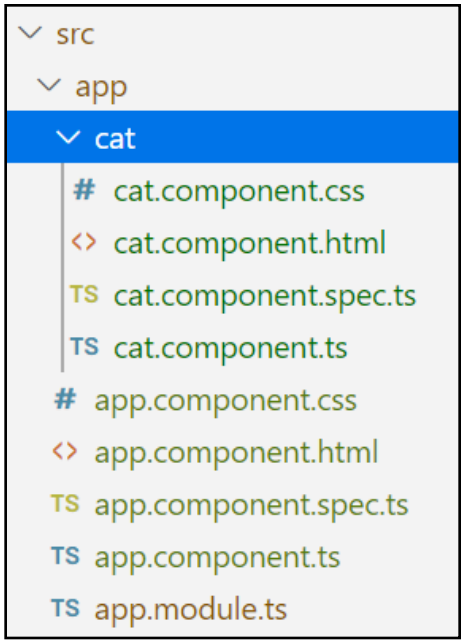
app.module.ts

```
ny-app > src > app > TS app.module.ts > ...
1   import { BrowserModule } from '@angular/platform-browser';
2   import { NgModule } from '@angular/core';
3   import { FormsModule } from '@angular/forms';
4   import { AppComponent } from './app.component';
5
6   @NgModule({
7     declarations: [
8       AppComponent
9     ],
10    imports: [
11      BrowserModule,
12      FormsModule
13    ],
```

❑ 2) template 기반 폼 검증

2) 자식 컴포넌트 생성

```
C:\angular\my-app>ng g component Cat
CREATE src/app/cat/cat.component.html (18 bytes)
CREATE src/app/cat/cat.component.spec.ts (607 bytes)
CREATE src/app/cat/cat.component.ts (263 bytes)
CREATE src/app/cat/cat.component.css (0 bytes)
UPDATE src/app/app.module.ts (446 bytes)
```



cat.component.html | app.module.ts ● | cat.component.ts | app.component.ht

app > app.module.ts > AppModule

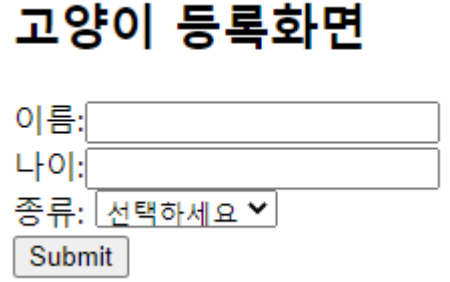
```ts
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { CatComponent } from './cat/cat.component';
import { FormsModule} from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent,
    CatComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
  ],
```

❑ 2) template 기반 폼 검증

3) 화면 레이아웃 꾸미기

cat.component.html

```
y-app > src > app > cat > <> cat.component.html > ...
 1   <h2>고양이 등록화면</h2>
 2   <form>
 3       이름:<input type="text" name="name" id="name" required>
 4       <br>
 5       나이:<input type="text" name="age" id="age" required>
 6       <br>
 7       종류:
 8       <select name="species" id="species">
 9           <option>선택하세요</option>
10           <option>페르시안</option>
11           <option>시암고양이</option>
12           <option>먼치킨</option>
13       </select><br>
14       <input type="submit" value="Submit">
15   </form>
```



cat.component.html | app.component.html

```
src > app > <> app.component.html > app-cat
 1   <app-cat></app-cat>
```

4) Submit 이벤트 및 데이터 저장 객체 생성

```
<h2>고양이 등록화면</h2>
<form (ngSubmit)="onSubmit()" >
    이름:<input type="text" name="name" id="name" required >
    <br>
    나이:<input type="text" name="age" id="age" required >
    <br>
```

```
export class CatComponent{

  cat ={
    name:'',
    age:'',
    species:'페르시안'
  }

  onSubmit(){
    console.log("onSubmit");
  }
```

❑ 2) template 기반 폼 검증

5) 저장 객체와 양방향 바인딩 및 #변수 이용한 alias 설정

cat.component.html

y-app > src > app > cat > cat.component.html > ...

```
1   <h2>고양이 등록화면</h2>
2   <form (ngSubmit)="onSubmit()" #catForm="ngForm">
3       이름:<input type="text" name="name" id="name" required [(ngModel)]="cat.name" #name="ngModel">
4       <br>
5       나이:<input type="text" name="age" id="age" required [(ngModel)]="cat.age" #age="ngModel">
6       <br>
7       종류:
8       <select name="species" id="species" [(ngModel)]="cat.species" #species="ngModel" >
9           <option>선택하세요</option>
10          <option>페르시안</option>
11          <option>시암고양이</option>
12          <option>먼치킨</option>
13      </select><br>
14      <input type="submit" value="Submit">
15  </form>
16  {{cat|json}}
```

6) required에 따른 유효성 검증값 출력하기

상태	의미
valid	검증이 유효한지에 대한 상태로서 검증을 통과했을 때 검증 결과가 true가 되고 반대의 경우에는 false가 됨
untouched	입력 상자에 포커스를 주지 않았을 경우 true가 되고 반대의 경우에는 false가 됨
touched	입력 상자에 포커스를 뒀다가 포커스를 잃어버린 경우, 터치했으므로 true가 되고 반대의 경우에는 false가 됨
pristine	값이 한 번도 입력되지 않았을 경우 비오염(pristine) 상태이기 때문에 true가 되고 오염 상태는 false가 됨
dirty	값이 한 번이라도 입력된 경우 더럽혀졌으므로 true 상태가 되고 반대의 경우에는 false가 됨

```
<h3>name 및 age 유효성 검증값</h3>
name.value={{name.value}}<br>
name.valid={{name.valid}}<br>
name.invalid={{name.invalid}}<br>
age.value={{age.value}}<br>
age.valid={{age.valid}}<br>
age.invalid={{age.invalid}}<br>
<h3>form 유효성 검증값</h3>
catForm.value={{catForm.value|json}}<br>
catForm.valid={{catForm.valid}}<br>
catForm.invalid={{catForm.invalid}}<br>
```

고양이 등록화면

이름:a
나이:b
종류: 시암고양이
Submit
{ "name": "a", "age": "b", "species": "시암고양이" }

name 및 age 유효성 검증값

name.value=a
name.valid=true
name.invalid=false
age.value=b
age.valid=true
age.invalid=false

form 유효성 검증값

catForm.value={ "name": "a", "age": "b", "species": "시암고양이
catForm.valid=true
catForm.invalid=false

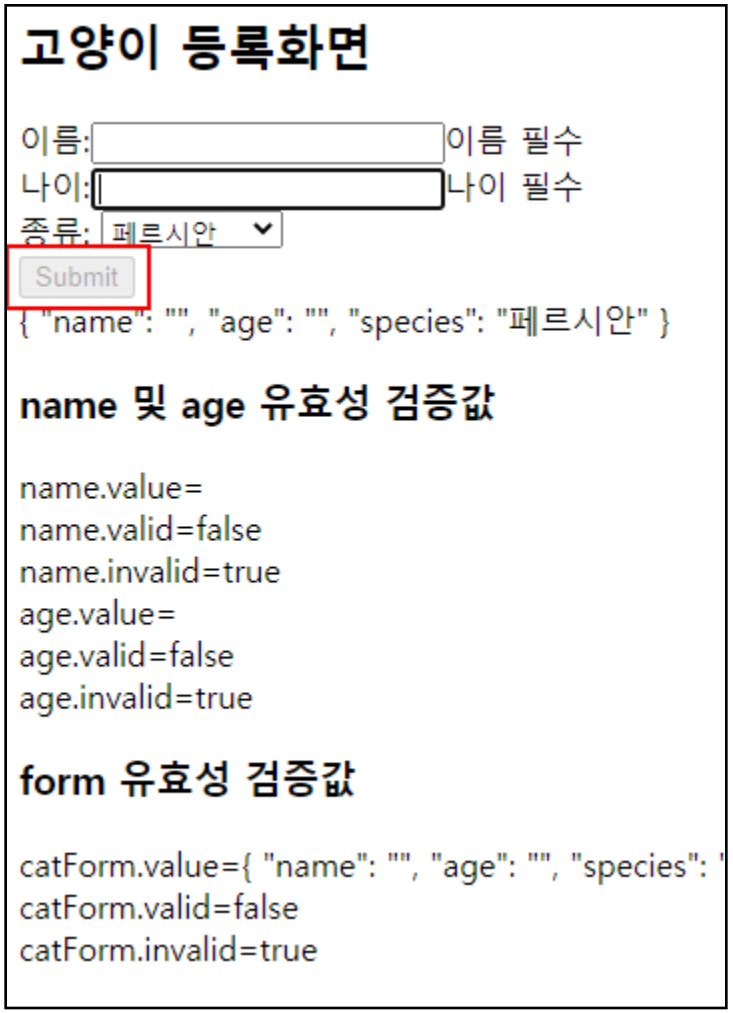
❑ 2) template 기반 폼 검증

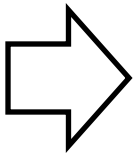
7) requried 위반시 경고 메시지 및 submit 버튼 비활성화

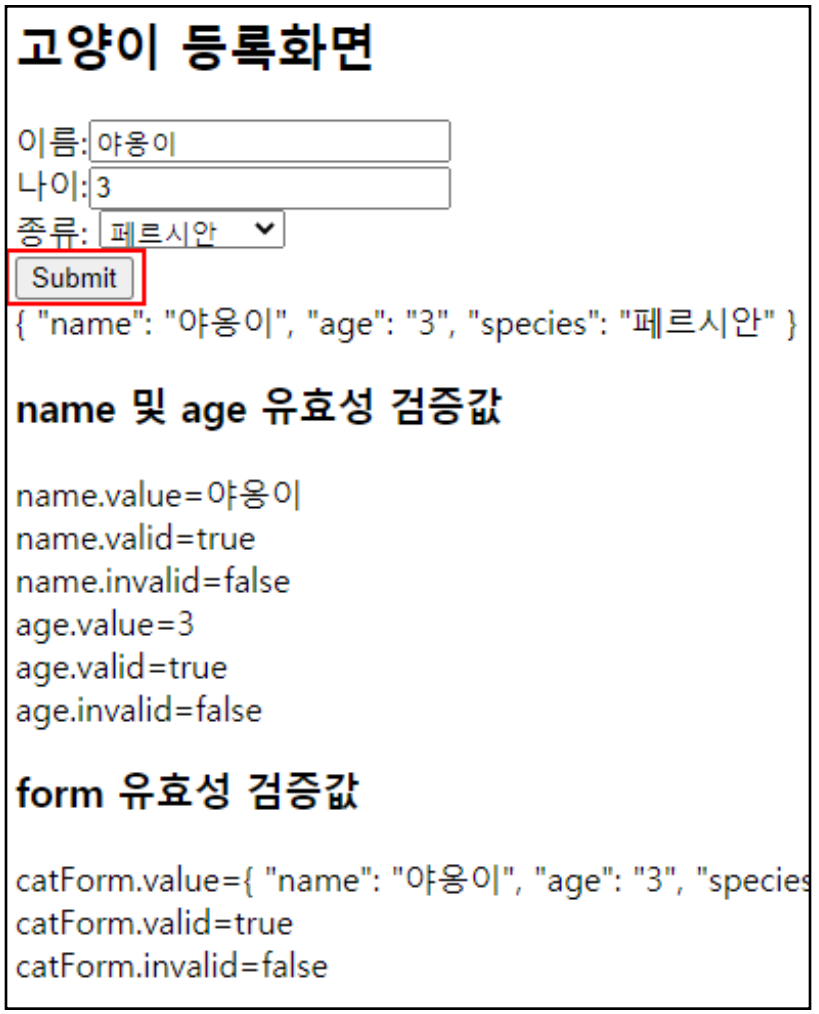
```
<h2>고양이 등록화면</h2>
<form (ngSubmit)="onSubmit()" #catForm="ngForm">
    이름:<input type="text" name="name" id="name" required [(ngModel)]="cat.name" #name="ngModel">
    <span [hidden]="name.valid">이름 필수</span>
    <br>
    나이:<input type="text" name="age" id="age" required [(ngModel)]="cat.age" #age="ngModel">
    <span *ngIf="age.invalid">나이 필수</span>
    <br>
    종류:
    <select name="species" id="species" [(ngModel)]="cat.species" #species="ngModel" >
        <option>선택하세요</option>
        <option>페르시안</option>
        <option>시암고양이</option>
        <option>먼치킨</option>
    </select><br>
    <input type="submit" value="Submit" [disabled]="catForm.invalid">
</form>
```

❑ 2) template 기반 폼 검증

8) 실행결과







❑ 3) model 기반 폼 검증

* model 기반

- template 내부에서 검증을 처리하지 않고 template 외부인 컴포넌트 클래스 내에서 선언한 모델에 검증을 위임하는 방식.

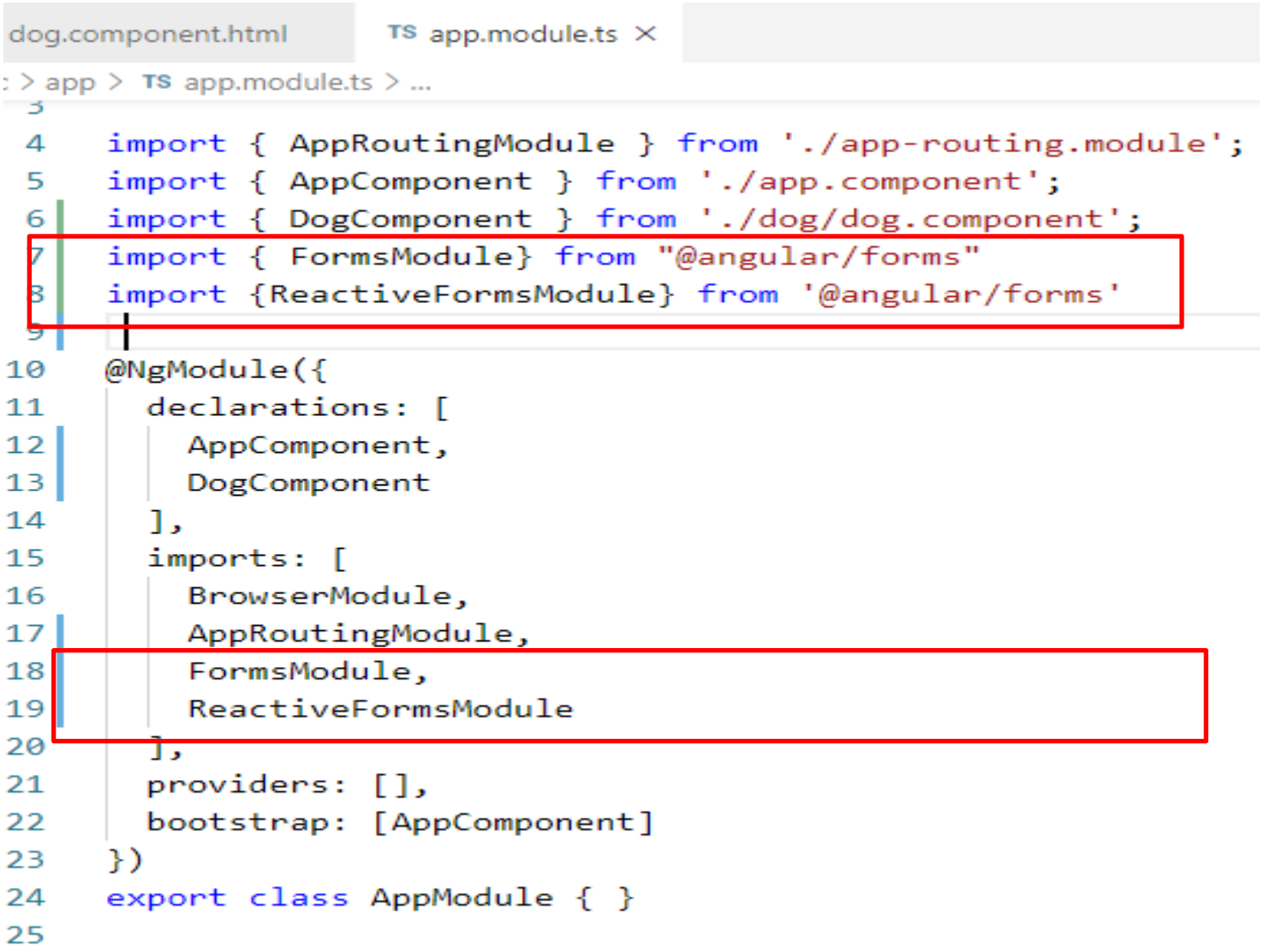
- FormGroup 클래스를 사용하며 ReactiveFormsModule 등록이 필요.

1) FormsModule 및 ReactiveFormsModule 모듈 등록

2) 자식 컴포넌트 생성

```
C:\angular\my-app>ng g component Dog
CREATE src/app/dog/dog.component.html (18 bytes)
CREATE src/app/dog/dog.component.spec.ts (607 bytes)
CREATE src/app/dog/dog.component.ts (263 bytes)
CREATE src/app/dog/dog.component.css (0 bytes)
UPDATE src/app/app.module.ts (447 bytes)
```

❑ formModule, ReactiveFormModule 등록



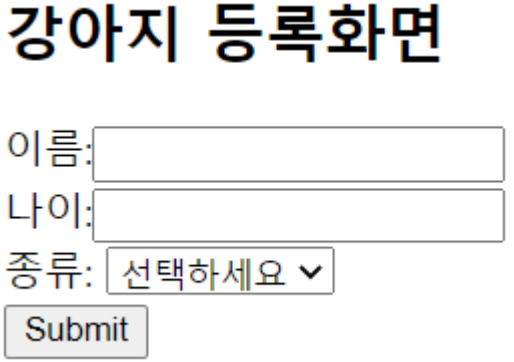
❑ 3) model 기반 폼 검증

3) 화면 레이아웃 꾸미기

dog.component.html

```
y-app > src > app > dog > <> dog.component.html > ...
```

```
<h2>강아지 등록화면</h2>
<form>
    이름:<input type="text" name="name" id="name" >
    <br>
    나이:<input type="text" name="age" id="age" >
    <br>
    종류:
    <select name="species" id="species">
        <option>선택하세요</option>
        <option>진도개</option>
        <option>푸들</option>
        <option>치와와</option>
    </select><br>
    <input type="submit" value="Submit">
</form>
```



❑ 3) model 기반 폼 검증

4) Submit 이벤트 및 데이터 저장 객체 생성

dog.component.html

y-app > src > app > dog > dog.component.html > ...

```
1   <h2>강아지 등록화면</h2>
2   <form (ngSubmit)="onSubmit()" >
3       이름:<input type="text" name="name" id="name" >
4       <br>
5       나이:<input type="text" name="age" id="age" >
6       <br>
7       종류:
8       <select name="species" id="species" >
9           <option>선택하세요</option>
10          <option>진도개</option>
11          <option>푸들</option>
12          <option>치와와</option>
13      </select><br>
14      <input type="submit" value="Submit">
15  </form>
16  {{dog|json}}
```

```
export class DogComponent{

  dog ={
    name:'',
    age:'',
    species:'진도개'
  }

  onSubmit(){
    console.log("onSubmit");
  }
```

5) 컴포넌트 클래스에서 FormGroup 객체 생성

```
export class DogComponent{

  ////////////////////////////////////////////////////
  form = new FormGroup({
    required_1: new FormControl('',Validators.required),
    upperCase_2: new FormControl('',
          Validators.compose([Validators.required,
                              Validators.pattern("[A-Z]{3}")]))
  });
  ////////////////////////////////////////////////////

  dog ={
    name:'',
    age:'',
    species:'진도개'
  }
```

- Validators.required: 값이 한 글자 이상인지를 검증함
- Validators.minLength(5): 최소 길이가 5이상인지를 검증함
- Validators.maxLength(5): 최대 길이가 5이하인지를 검증함
- Validators.pattern("[a-zA-Z]+"): 정규식(알파벳으로만 이뤄졌는지)을 검증함

6) formGroup 및 formControl를 template내 등록

dog.component.html

y-app > src > app > dog > dog.component.html > ...

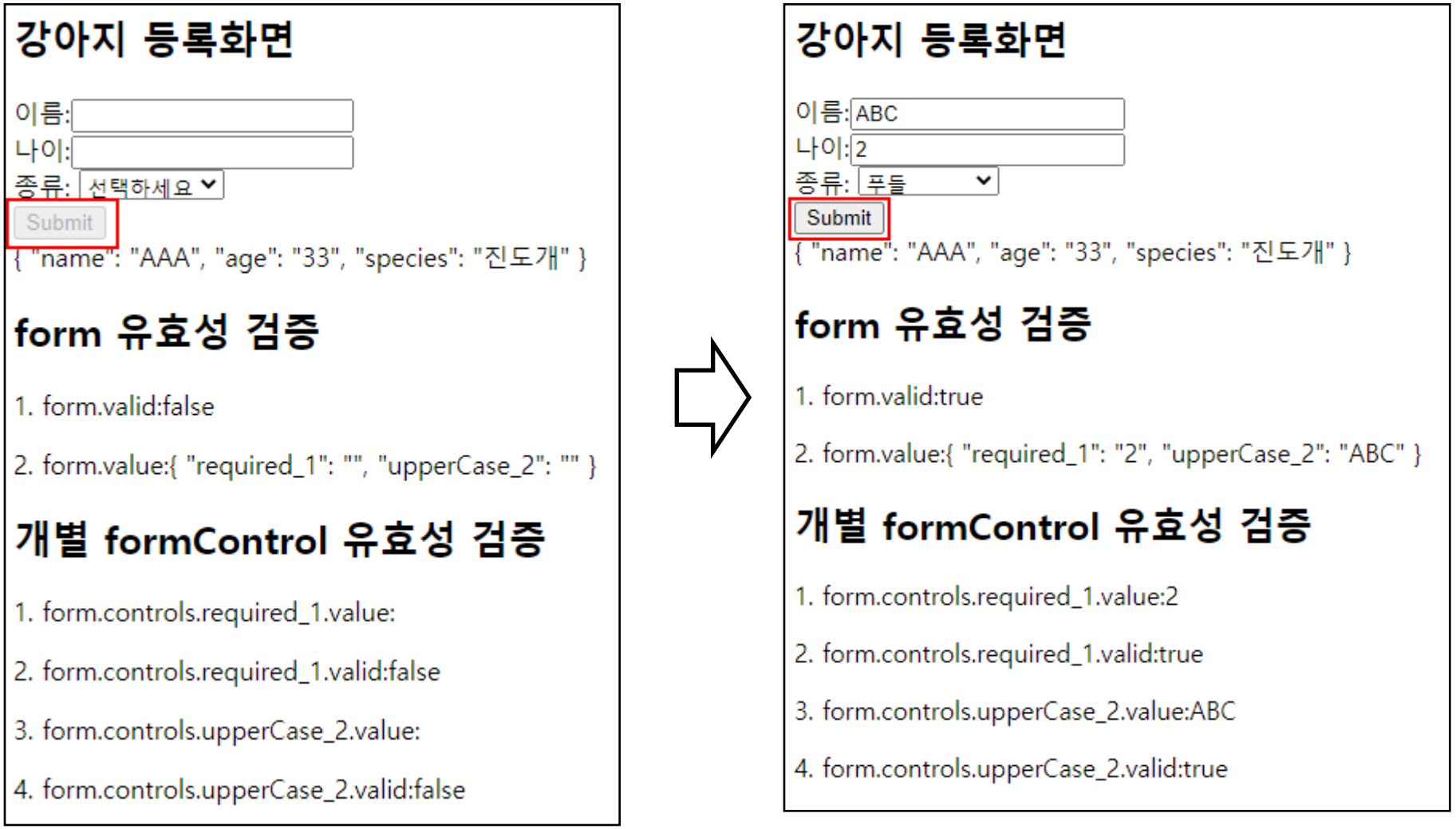
```
<h2>강아지 등록화면</h2>
<form (ngSubmit)="onSubmit()" [formGroup]="form" >
    이름:<input type="text" name="name" id="name" formControlName="upperCase_2">
    <br>
    나이:<input type="text" name="age" id="age"  formControlName="required_1">
    <br>
    종류:
    <select name="species" id="species" >
        <option>선택하세요</option>
        <option>진도개</option>
        <option>푸들</option>
        <option>치와와</option>
    </select><br>
    <input type="submit" value="Submit">
</form>
{{dog|json}}
```

❑ 3) model 기반 폼 검증

7) FormControl 위반시 경고 메시지 및 submit 버튼 비활성화

```
<h2>강아지 등록화면</h2>
<form (ngSubmit)="onSubmit()" [formGroup]="form" >
    이름:<input type="text" name="name" id="name" formControlName="upperCase_2">
    <br>
    나이:<input type="text" name="age" id="age"  formControlName="required_1">
    <br>
    종류:
    <select name="species" id="species" >
        <option>선택하세요</option>
        <option>진도개</option>
        <option>푸들</option>
        <option>치와와</option>
    </select><br>
    <input type="submit" value="Submit" [disabled]="form.invalid">
</form>
{{dog|json}}
<h2>form 유효성 검증</h2>
<p>1. form.valid:{{form.valid}}</p>
<p>2. form.value:{{form.value|json}}</p>
<h2>개별 formControl 유효성 검증</h2>
<p>1. form.controls.required_1.value:{{form.controls.required_1.value}}</p>
<p>2. form.controls.required_1.valid:{{form.controls.required_1.valid}}</p>
<p>3.  form.controls.upperCase_2.value:{{form.controls.upperCase_2.value}}</p>
<p>4. form.controls.upperCase_2.valid:{{form.controls.upperCase_2.valid}}</p>
```

8) 실행 결과


강아지 등록화면
이름:
나이:
종류: 선택하세요
Submit
{ "name": "AAA", "age": "33", "species": "진도개" }
form 유효성 검증
1. form.valid:false
2. form.value:{ "required_1": "", "upperCase_2": "" }
개별 formControl 유효성 검증
1. form.controls.required_1.value:
2. form.controls.required_1.valid:false
3. form.controls.upperCase_2.value:
4. form.controls.upperCase_2.valid:false
강아지 등록화면
이름:ABC
나이:2
종류: 푸들
Submit
{ "name": "AAA", "age": "33", "species": "진도개" }
form 유효성 검증
1. form.valid:true
2. form.value:{ "required_1": "2", "upperCase_2": "ABC" }
개별 formControl 유효성 검증
1. form.controls.required_1.value:2
2. form.controls.required_1.valid:true
3. form.controls.upperCase_2.value:ABC
4. form.controls.upperCase_2.valid:true

❑ 사용자 정의 Validator생성

```
C:\angular_chul\chul-app>ng g class CustomValidator
CREATE src/app/custom-validator.spec.ts (191 bytes)
CREATE src/app/custom-validator.ts (33 bytes)

C:\angular_chul\chul-app>
```

```
C:\angular_chul\chul-app>ng g component dog
CREATE src/app/dog/dog.component.html (18 bytes)
CREATE src/app/dog/dog.component.spec.ts (605 bytes)
CREATE src/app/dog/dog.component.ts (263 bytes)
CREATE src/app/dog/dog.component.css (0 bytes)
UPDATE src/app/app.module.ts (570 bytes)
```

app.module.ts - chul-app - Visual Studio Code

custom-validator.ts | app.module.ts ×

src > app > app.module.ts > AppModule

```
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import {FormsModule, ReactiveFormsModule} from '@angular/forms';
7
8  @NgModule({
9    declarations: [
10     AppComponent
11   ],
12   imports: [
13     BrowserModule,
14     AppRoutingModule,
15     FormsModule,
16     ReactiveFormsModule
17   ],
```

❑ 4) 커스텀 Validator 생성

* 커스텀 유효성 검증

- 기본적으로 Angular에서 제공하는 내장 방법으로 검증.
- 필요시 사용자가 직접 유효성 검증을 위한 Validator 객체 생성 가능.

- static 메서드 및 FormControl 인자타입, 반환값은 { 메서드명:true} 형식 사용

custom-validator.ts

y-app > src > app > TS custom-validator.ts > ...

```
1   import { FormControl } from '@angular/forms';
2
3   export class CustomValidator {
4
5       static eq5(control:FormControl){
6           var result = null;
7           if(control.value != 5){
8               result =  {eq5:true} // 유효성 검증에 문제 있음
9           }
10          return result;  // null이면 문제없음 의미
11      }
12  }
```

❑ 4) 커스텀 Validator 생성

FormGroup 객체에 등록

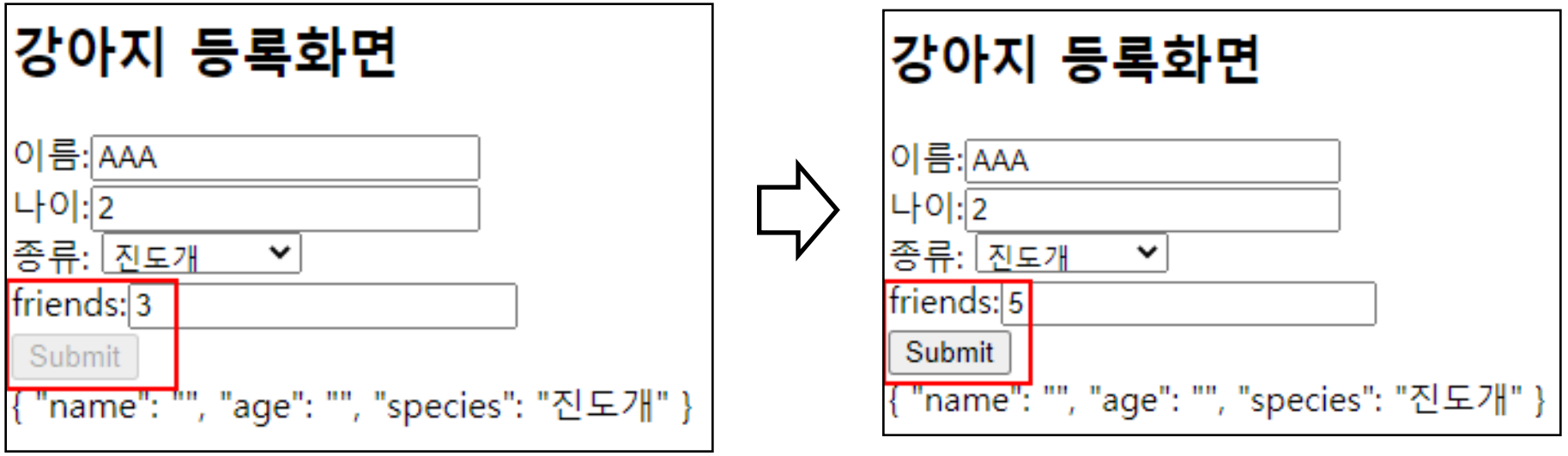
```
export class DogComponent{

  ////////////////////////////////////////////////
  form = new FormGroup({
    required_1: new FormControl('',Validators.required),
    eq5_1: new FormControl('', CustomValidator.eq5),
    upperCase_2: new FormControl('',
                Validators.compose([Validators.required,
                                    Validators.pattern("[A-Z]{3}")]))
  });
   ////////////////////////////////////////////////
```

```
종류:
<select name="species" id="species">
    <option>선택하세요</option>
    <option>진도개</option>
    <option>푸들</option>
    <option>치와와</option>
</select><br>
friends:<input type="text" name="friends" id="friends"  formControlName="eq5_1">
<input type="submit" value="Submit" [disabled]="form.invalid">
```

❑ 사용자 정의 Validator의 사용

```
10  })
11  export class DogComponent implements OnInit {
12  ////////////////////////////////////
13  form = new FormGroup({
14    required_1:new FormControl('', Validators.required),
15    eq5_1:new FormControl('',CustomValidator.eq5 ),//사용자정의 validator사용
16    upperCase_2:new FormControl('',
17    Validators.compose([Validators.required,
18    Validators.pattern("[A-Z]{3}")]))
19  });
20
```

❑ 4) 커스텀 Validator 생성



수고하셨습니다.