This page will walk through Angular 2 FormGroup example. FormGroup takes part in creating reactive form. FormGroup is used with FormControl and FormArray. The role of FormGroup is to track the value and validation state of form control. In our example we will create a form that will include <input> and <select> element. We will see how to instantiate FormGroup and how to set and access values in FormGroup. We will also validate the form. First let us understand the angular classes and directive that we will use in our example.   
  
**FormControl**:It is a class that tracks the value and validation state of a form control.   
**FormGroup**: It is a class that tracks the value and validity state of a group of FormControl.   
**FormArray**: It is a class that tracks the value and validity state of array of FormControl, FormGroup and FormArray.  
**FormControlName**: It is a directive that syncs a FormControl in an existing FormGroup to a form control by name.   
**FormGroupDirective**: It is a directive that binds an existing FormGroup to a DOM element.   
**FormGroupName**: It is a directive that syncs a nested FormGroup to a DOM element.   
**FormArrayName**: It is a directive that syncs a nested FormArray to a DOM element.

FormControl

FormControl is used to track values and validation of a form control. It can be used standalone as well as with the parent form. When we work with FormControl class, FormControlDirective and FormControlName directives are also used.   
**FormControlDirective**: It syncs a standalone FormControl instance to form a control element.   
**FormControlName**: It is used with FormGroup with a <form> tag. FormControlName syncs a FormControl in an existing FormGroup to a form control by name.   
  
Now let us start how to use FormControl in our angular application. For every form control such as text, checkbox, radio button we need to create an instance of FormControl in our class.

city = new FormControl();

In our HTML template it will be used as

<input [formControl]="city">

Here [formControl] is FormControlDirective. Find the example   
**formcontrol.component.ts**

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

import { FormControl, Validators } from '@angular/forms';

@Component({

selector: 'app-form-control',

templateUrl: 'formcontrol.component.html',

styleUrls: ['formcontrol.component.css']

})

export class FormControlDemoComponent {

name = new FormControl('', [Validators.required, Validators.maxLength(15)]);

age = new FormControl(20, Validators.required);

city = new FormControl();

country = new FormControl({value: 'India', disabled: true});

married = new FormControl(true);

setNameValue() {

this.name.setValue('Donald Trump');

console.log('Name: ' + this.name.value);

console.log('Validation Status: ' + this.name.status);

}

setResetName() {

this.name.reset();

}

changeValue() {

console.log(this.married.value);

this.married = new FormControl(!this.married.value);

}

}

In Angular 4, the following four statuses are commonly used by forms:

* valid – state of the validity of all form controls, true if all controls are valid
* invalid – inverse of valid; true if some control is invalid
* pristine – gives a status about the “cleanness” of the form; true if no control was modified
* dirty – inverse of pristine; true if some control was modified

There are two ways in which you can use the ActivatedRoute to get the parameter value

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Example for Template-Driven Forms**

<h3>Template Driven Form</h3>

<p \*ngIf="isValidFormSubmitted" [ngClass] = "'success'">

Form submitted successfully.

</p>

<form #userForm="ngForm"

(ngSubmit)="onFormSubmit(userForm)">

<table>

   <tr>

   <td>Name:</td>

   <td>

         <input name="uname" [ngModel]="user.userName"

         required #uname="ngModel">

   </td>

   </tr>

   <tr>

   <td>Gender:</td>

   <td>

       <input type="radio" value="male" name="gender"

        [ngModel]="user.gender" required

        #gender="ngModel">

         Male

         <input type="radio" value="female"

          name="gender" [ngModel]="user.gender"

          required #gender="ngModel"> Female

    </td>

   </tr>

   <tr>

   <td>Are you married? </td>

   <td>

     <input type="checkbox" name="married"

     [ngModel]="user.isMarried">

   </td>

   </tr>

   <tr>

   <td>Accept T & C </td>

   <td>

   <input type="checkbox" name="tc" [ngModel]="user.isTCAccepted" required #tc="ngModel">

   </td>

   </tr>

   <tr>

   <td colspan="2">

   <button>Submit</button>

     <button type="button" (click)="setDefaultValues()">Set Default</button>

     <button type="button" (click)="resetForm(userForm)">Reset</button>

   </td>

   </tr>

</table>

</form>

import { Component, OnInit } from '@angular/core';

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

import { UserService } from './services/user-service';

import { User } from './domain/user';

@Component({

selector: 'app-template',

templateUrl: './template-driven-form.component.html',

styleUrls: ['./style.css']

})

export class TemplateDrivenFormComponent {

isValidFormSubmitted = false;

    user = new User();

    constructor(private userService: UserService) {

    }

    onFormSubmit(form: NgForm) {

     this.isValidFormSubmitted = false;

     if(form.invalid){

         return;

     }

     this.isValidFormSubmitted = true;

     this.user.userName = form.controls['uname'].value;

     this.user.gender = form.controls['gender'].value;

     this.user.isMarried = form.controls['married'].value;

     this.user.isTCAccepted = form.controls['tc'].value;

     this.userService.createUser(this.user);

     this.resetForm(form);

    }

    resetForm(form: NgForm) {

     form.resetForm({

         married: false

     });

    }

    setDefaultValues() {

     this.user.userName = 'Krishna';

     this.user.gender = 'male';

     this.user.isMarried = true;

     this.user.isTCAccepted = false;

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Reactive Form**

import { Component, OnInit } from '@angular/core';

import { FormControl, FormGroup, Validators }

from '@angular/forms';

import { UserService } from './services/user-service';

import { User } from './domain/user';

@Component({

selector: 'app-reactive',

templateUrl: './reactive-form.component.html',

styleUrls: ['./style.css']

})

export class ReactiveFormComponent {

isValidFormSubmitted: boolean = null;

    userForm = new FormGroup({

     uname: new FormControl('', Validators.required),

     gender: new FormControl('', Validators.required),

     married: new FormControl(false),

     tc: new FormControl('', Validators.required)

    });

    user = new User();

    constructor(private userService: UserService) {

    }

onFormSubmit() {

     this.isValidFormSubmitted = false;

     if(this.userForm.invalid){

         return;

     }

     this.isValidFormSubmitted = true;

     console.log(this.userForm.valid);

     this.user.userName = this.userForm.get('uname').value;

     this.user.gender = this.userForm.get('gender').value;

     this.user.isMarried = this.userForm.get('married').value;

     this.user.isTCAccepted = this.userForm.get('tc').value;

     this.userService.createUser(this.user);

     this.reset();

    }

    reset() {

     this.userForm.reset({

         married: false

     });

    }

    setDefaultValues() {

     this.userForm.patchValue({uname: 'Krishna', gender:'male', married:true});

    }

}

<h3>Reactive Form</h3>

<p \*ngIf="isValidFormSubmitted && userForm.pristine"

[ngClass] = "'success'">

Form submitted successfully.

</p>

<form [formGroup]="userForm" (ngSubmit)="onFormSubmit()">

<table>

   <tr>

   <td>Name:</td>

   <td>

   <input formControlName="uname">

       <div \*ngIf="userForm.get('uname').invalid &&

        isValidFormSubmitted != null &&

        !isValidFormSubmitted" [ngClass] = "'error'">

       Name required.

     </div>

   </td>

   </tr>

   <tr>

   <td>Gender:</td>

   <td>

       <input type="radio" value="male"

       formControlName="gender"> Male

         <input type="radio" value="female"

         formControlName="gender"> Female

     <div \*ngIf="userForm.get('gender').invalid &&

     isValidFormSubmitted != null &&

     !isValidFormSubmitted" [ngClass] = "'error'">

       Gender required.

     </div>

    </td>

   </tr>

   <tr>

   <td>Are you married? </td>

   <td>

   <input type="checkbox" formControlName="married">

   </td>

   </tr>

   <tr>

   <td>Accept T & C </td>

   <td>

   <input type="checkbox" formControlName="tc">

     <div \*ngIf="userForm.get('tc').invalid && isValidFormSubmitted != null && !isValidFormSubmitted" [ngClass] = "'error'">

     Accept T & C.

     </div>

   </td>

   </tr>

   <tr>

   <td colspan="2">

   <button>Submit</button>

     <button type="button" (click)="setDefaultValues()">Set Default</button>

     <button type="button" (click)="reset()">Reset</button>

   </td>

   </tr>

</table>

</form>