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

1. Using Snapshot
2. Using observable

**Using Snapshot**

C#

|  |  |
| --- | --- |
|  | this.id=this.\_Activatedroute.snapshot.params['id']; |

The snapshot property returns initial value of the route. You can then access the params array, to access the value of the id as shown above

Use this option, if you only need the initial value.

**Using Observable**

C#

|  |  |
| --- | --- |
|  | \_Activatedroute.params.subscribe(params => { this.id = params['id']; }); |

You can retrieve the value of id by subscribing to the params observable property of the activateRoute as shown above

Use this option if you expect the value of the parameter to change over time.

**Why use observable**

We usually retrieve the value of the parameter in the **ngOninit** life cycle hook, when the component initialised.

When the user navigates to the component again, the Angular does not create the new component but reuses the existing instance. In such circumstances, the ngOnInit method of the component is not called again. Hence you need a way to get the value of the parameter.

By subscribing to the observable params property, you will retrieve the latest value of the parameter and update the component accordingly.

The above difference is explained in our next tutorial Angular child routes tutorial.