

*Universidad Tecnológica de la*

*Riviera Maya*

*Solidaridad, Quintana Roo, 01 de febrero del 2018*

**Desarrollos de aplicaciones III**

*ngForm*

**Profesor:**

*Julián Cristóbal Alonzo Villegas*

**Autor(es):**

*Luis Fernando Morales Maldonado*

**Carrera:**

*TSU en Tecnologías de la Información y Comunicación Área Sistemas Informáticos (TI51)*

Contenido

[ngForm 2](#_Toc510734278)

[Información de la directiva 2](#_Toc510734279)

[Argumentos 3](#_Toc510734280)

[Implementación 3](#_Toc510734281)

[Directiva NgForm 3](#_Toc510734282)

[NgModel Directive 6](#_Toc510734283)

[NgForm con NgModel 6](#_Toc510734284)

[Ejemplo completo 9](#_Toc510734285)

[Referencias 12](#_Toc510734286)

[Fuente de information 12](#_Toc510734287)

# ngForm

Alias ​​identificable de la [form](https://docs.angularjs.org/api/ng/directive/form)directiva. HTML no permite la anidación de elementos de formulario. Es útil anidar formularios, por ejemplo, si se debe determinar la validez de un subgrupo de controles.

Nota: el propósito de ngFormes agrupar controles, pero no ser un reemplazo para la <form>etiqueta con todas sus capacidades (por ejemplo, publicar en el servidor.

NgFormdirectiva se utiliza con la etiqueta de formulario HTML que se puede exportar en la variable de plantilla local para acceder a los valores de formulario y estado de validación, y para pasar el formulario completo a nuestra clase en el envío de formulario. NgModelse usa en los campos del formulario y se puede exportar en una variable de plantilla local para obtener sus valores y el estado de validación en la plantilla HTML. Cuando creamos un formulario HTML, debemos usar el nameatributo en los campos de formulario. En el contexto de NgForm, el rol de NgModeles registrar el campo de formulario con formulario usando el nameatributo.NgModeltambién se usa para enlazar el modelo de dominio de la clase con view.

## Información de la directiva

* Esta directiva se ejecuta en el nivel de prioridad 0

Uso

* como elemento:
* <ng-form
* [name="string"]>
* ...

</ng-form>

* como atributo:
* <ANY
* [ng-form="string"]>
* ...

</ANY>

* como clase de CSS:

<ANY class="[ng-form: string;]"> ... </ANY>

## Argumentos

| **Param** | **Tipo** | **Detalles** |
| --- | --- | --- |
| ngForm |nombre  *(Opcional)* | [**cuerda**](https://docs.angularjs.org/) | Nombre del formulario Si se especifica, el controlador de formulario se publicará en el alcance relacionado, bajo este nombre. |

# Implementación

## Directiva NgForm

NgFormdirectiva se usa con <form>etiqueta HTML . Para entender NgForm, debemos tener cuidado con las siguientes clases.   
**FormControl** : esta clase rastrea el valor y el estado de validación de un control de formulario individual.   
**FormGroup** : esta clase rastrea el valor y el estado de validez de un grupo de FormControl.   
  
Ahora déjenos entender la NgFormdirectiva. NgFormes una directiva que crea una FormGroupinstancia de nivel superior y la vincula a un formulario para rastrear el valor agregado del formulario y el estado de validación.   
  
Encuentre los pasos para usar NgFormen nuestra aplicación.   
**1.** Configure FormsModuleen los importsmetadatos del @NgModuledecorador en el módulo de la aplicación como se indica a continuación.

importar { FormsModule } desde '@ angular / forms' ;

@NgModule ({ --------------- ---------------

imports : [ BrowserModule , FormsModule ] --------- ------ --------------- })

**2.** Una vez que importamos FormsModule, la NgFormdirectiva se activa por defecto en todas las <form>etiquetas. Podemos exportar NgForma una variable de plantilla local. Para importarlo necesitamos usar NgFormcomo ngForm. Encuentre el ejemplo de muestra.

<form #userForm="ngForm" (ngSubmit)="onFormSubmit(userForm)">

----------------

----------------

<button>Submit</button>

</form>

Para enviar el formulario podemos usar el <button>tipo que se **envía** de forma predeterminada para todo el navegador moderno.   
La exportación NgForma la variable de plantilla local es opcional pero útil. En el fragmento de código anterior, fíjate en #userForm="ngForm". Aquí la variable de plantilla local es userForm. Usando la variable de plantilla local podemos acceder al formulario en nuestra plantilla HTML de la siguiente manera.   
**userForm.value** : **Devolverá el** valor del formulario. Significa que obtenemos un objeto que contiene los valores de todos los campos utilizados en la <form>etiqueta.   
**userForm.valid** : **Devolverá el** valor booleano. Cuando el valor es **verdadero** , la forma es válida. Se dice que el formulario es válido si todos los campos devuelven la validez **verdadera** .   
**userForm.touched**: Devolverá el valor booleano. Si el valor es **verdadero** , significa que el usuario ha ingresado valor al menos en un campo.   
**userForm.submitted** : **Devolverá el** valor booleano. Si el valor es **verdadero** , significa que la forma ha sido enviada.   
  
Usamos la variable de plantilla local para enviar datos de formulario. Examinar

( ngSubmit ) = "onFormSubmit (userForm)"

Aquí, cuando se envía el formulario, se pasa una variable de plantilla local a la que se puede acceder en clase de la siguiente manera.

onFormSubmit(userForm: NgForm) {

console.log(userForm.value);

console.log('Name:' + userForm.controls['name'].value);

console.log('Form Valid:' + userForm.valid);

console.log('Form Submitted:' + userForm.submitted);

}

Usando controls[]de NgForm, podemos acceder al valor del campo de formulario por nombre de campo en nuestra clase de la siguiente manera.

userForm . controles [ 'nombre' ]. valor ;

userForm . controla [ 'ciudad' ]. valor ;

userForm . controles [ 'estado' ]. valor

Aquí el nombre, la ciudad, el estado son los nombres de los campos.   
  
**3.** Para restablecer el formulario, NgFormtiene el resetForm()método que se llama de la siguiente manera.

resetUserForm ( userForm : NgForm ) {

userForm . resetForm () ;; }

Para llamar a la función anterior, crea un botón en la interfaz de usuario.

<button type = "button" ( click ) = "resetUserForm (userForm)" > Restablecer </ button>

Si queremos restablecer el formulario con algunos valores predeterminados, entonces asigne los valores predeterminados con el nombre de control de formulario del formulario.

resetUserForm ( userForm : NgForm ) {

userForm . resetForm ({

name : 'Mahesh' ,

ciudad : 'Noida' }); }

## NgModel Directive

NgModel directive creates a FormControl instance from a domain model created in class and binds it to a form control element. FormControl keeps the track of user interaction and validation status of the form control. NgModelcan be used standalone as well as with the parent form. When it is used as standalone, we can use it as follows.

<input [(ngModel)]="name" required #userName="ngModel">

<p>Name value: {{ name }} </p>

<p>Name value: {{ userName.value }} </p>

<p>Name valid: {{ userName.valid }} </p>

Now look into #userName="ngModel". Here userName is local template variable that can be used to get form control value, valid, invalid, pending etc properties in HTML template.   
NgModel can be used as one-way binding and two-way binding. In one way binding using **[ ]** sign, when value changes in domain model in the class then that is set to view also. In two-way binding using **[( )]** sign, the value changes in domain model is set to view and values changes in view is also set to domain model.

### NgForm con NgModel

Usaremos aquí NgFormcon la NgModeldirectiva. Cuando usamos ngModella <form>etiqueta, también tenemos que proporcionar un nameatributo para registrar un control con forma parental usando ese nombre.   
Cuando estamos usando el formulario padre, no necesitamos usar el enlace bidireccional para obtener valores de campos de formulario en nuestra clase. Podemos pasar <form>una variable de plantilla local, como userFormnuestro método de envío, tal onFormSubmit(userForm)como se indica a continuación.

<form # userForm = "ngForm" ( ngSubmit ) = "onFormSubmit (userForm)" >

En el formulario enviado, se puede acceder a los valores del formulario en clase como se indica a continuación.

onFormSubmit ( userForm : NgForm ) {

consola . log ( userForm . value );

consola . log ( 'Nombre:' + userForm . controls [ 'nombre' ]. valor ); }

Si no usamos el enlace unidireccional ni el enlace bidireccional, aun así necesitamos usar el ngModelatributo en los campos cuando lo usemos con el formulario principal como se indica a continuación.

<input name = "message" ngModel >

Si no usaremos el ngModelatributo en campos tales como

<input name = "message" >

Entonces este campo no se registrará con el formulario principal y su valor no se transferirá a nuestra clase en el envío de formularios.   
  
Vamos a entender caso por caso.   
**Caso-1:** Aquí estamos usando enlace bidireccional en forma primaria.

<form # userForm = "ngForm" ( ngSubmit ) = "onFormSubmit (userForm)" >

<input name = "message" [( ngModel )] = "msg" >

<botón> Enviar </ button> </ form>

Si queremos formar valores de campos usando el modelo de dominio, entonces podemos usar el enlace bidireccional. En el formulario enviado también podemos acceder a los valores del formulario usando NgFormen nuestra clase. En el caso de la forma parental, la mayoría de las veces, no necesitamos usar un enlace bidireccional porque podemos acceder a los valores en nuestra clase utilizando NgFormsolo. nameatributo es requerido para registrar el control con la forma.   
**Caso-2:** Aquí estamos usando el enlace unidireccional en forma primaria.

<form # userForm = "ngForm" ( ngSubmit ) = "onFormSubmit (userForm)" >

<input name = "message" [ ngModel ] = "msg" >

<botón> Enviar </ button> </ form>

Deberíamos usar el enlace unidireccional en el escenario donde queremos rellenar previamente los campos del formulario. nameatributo es requerido para registrar el control con la forma.   
**Caso-3:** Aquí no estamos usando un enlace bidireccional o bidireccional.

<form # userForm = "ngForm" ( ngSubmit ) = "onFormSubmit (userForm)" >

<input name = "message" ngModel >

<botón> Enviar </ button> </ form>

Para registrar el control secundario con el formulario, debemos usarlo ngModelcomo atributo de campo. Si no utilizamos el enlace unidireccional ni el enlace bidireccional, aún debemos usar el ngModelatributo para que el campo pueda registrarse con el formulario principal, de lo contrario, el valor del campo no se transmitirá en el envío de formulario. nameatributo es requerido para registrar el control con la forma.   
**Caso-4:** Aquí no estamos usando ngModelde ninguna manera.

<form # userForm = "ngForm" ( ngSubmit ) = "onFormSubmit (userForm)" >

<input name = "message" >

<botón> Enviar </ button> </ form>

Como sabemos, para registrar el control secundario con el formulario, debemos usarlo ngModelcomo atributo de campo. Si no estamos usando ngModelcomo atributo de campo, entonces ese campo no se registrará con el formulario principal y, por lo tanto, el valor del campo no se transmitirá a través del formulario enviado a nuestra clase.

## Ejemplo completo

**app.component.html**

<h3>Using NgForm with NgModel</h3>

<div>

<form #userForm="ngForm" (ngSubmit)="onFormSubmit(userForm)">

<div>

Name: <input name="name" ngModel required #userName="ngModel">

</div>

<div>

City: <input name="city" ngModel #userCity="ngModel">

</div>

<div>

State: <input name="state" ngModel>

</div>

<div>

<button>Submit</button>

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

</div>

</form>

</div>

<div>

<p>Form value: <b> {{ userForm.value | json }} </b></p>

<p>Form valid: <b> {{ userForm.valid }} </b></p>

<p>Form touched: <b> {{ userForm.touched }} </b></p>

<p>Form submitted: <b> {{ userForm.submitted }} </b></p>

</div>

<div>

<p>Name value: <b> {{ userName.value }} </b></p>

<p>Name valid: <b> {{ userName.valid }} </b></p>

</div>

<div>

<p>City value: <b> {{ userCity.value }} </b></p>

<p>City valid: <b> {{ userCity.valid }} </b></p>

</div>

**app.component.ts**

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

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

@Component({

selector: 'app-root',

templateUrl: 'app.component.html'

})

export class AppComponent {

onFormSubmit(userForm: NgForm) {

console.log(userForm.value);

console.log('Name:' + userForm.controls['name'].value);

console.log('Form Valid:' + userForm.valid);

console.log('Form Submitted:' + userForm.submitted);

}

resetUserForm(userForm: NgForm) {

userForm.resetForm();;

}

}

**app.module.ts**

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

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

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

@NgModule({

imports: [

BrowserModule,

FormsModule

],

declarations: [

AppComponent,

],

bootstrap: [

AppComponent

]

})

export class AppModule { }

Test Application

Para probar la aplicación, encuentre los siguientes pasos.   
**1** .: Descargue el código fuente utilizando el enlace de descarga que se proporciona en esta página.   
**2** .: en nuestra aplicación CLI angular, reemplace la carpeta **src** .   
**3** .: Ejecute el comando **ng serve** .   
**4** .: Ahora acceda a la URL **http: // localhost: 4200** . Encuentra la pantalla de impresión.



### Referencias

[NgForm](https://angular.io/docs/ts/latest/api/forms/index/NgForm-directive.html)   
[NgModel](https://angular.io/docs/ts/latest/api/forms/index/NgModel-directive.html)

# Fuente de information

[angular-2-ngform-with-ngmodel-directive-example.zip](https://www.concretepage.com/angular-2/download/angular-2-ngform-with-ngmodel-directive-example.zip)

<https://www.concretepage.com/angular-2/angular-2-ngform-with-ngmodel-directive-example>

https://docs.angularjs.org/api/ng/directive/ngForm