Create **Angular** CLI Project

Let's start to create a **new Angular application using CLI**[At the time of writing this article, we are using **Angular** 6].

1. Open **Visual Studio Code** and open terminal windows to press **Ctrl+`**
2. Execute command **ng new AngularCRUDExample --routing** and press Enter
3. Execute command **ng serve --open** to run your project.
4. Stop the running project using**Ctrl+C.**
5. Execute command **npm install bootstrap@3 --save** for installing bootstrap.
6. Execute command **npm install glyphicons --save** for installing glyphicons.
7. Execute command**npm install ngx-bootstrap --save** for installing ngx-bootstrap which provide module like Modal, Tabs etc.

**CRUD** **Operation**s Implementation

Now, let's move to the next step and create one service first before any other implementation, where we will define all the **CRUD** **operation**s function. So, first create a folder name with '**Services**' and from the terminal windows move to this folder path and execute the following command for creating a service name with '**BlogService**'.

**ng generate service Blog**

Now going to create all methods which are required for **CRUD** **operation**s.

* First, let import **HttpClient**and **HttpHeaders** from **@angular/common/http** [Don't forget to import HttpClientModule on AppModule].
* As we will use live API which is running on IIS, first let configure the base URL for the API service.
* Then create the list of methods which are essential for **CRUD** **operation**s like
  + **getCategoryList()**: To get the list of available category.
  + **getPostList()**: To get the list of available blog post.
  + **addPost()**: For adding new post
  + **deletePost()**: For deleting existing post
  + **updatePost()**: For updating existing post
  + **getPost()**: To get individual post details
  + **changePostId()**: This is special method which will use to get the post id while updating the existing post.
* **Don't forget to add the header with Content-Type.**

**blog.service.ts**

Hide   Shrink Image 1 for CRUD Operations in Angular with Typescript using ngx-bootstrap  Copy **Code**

import { Injectable } from "@**angular**/core";

import { HttpClient, HttpHeaders } from "@**angular**/common/http";

import { BehaviorSubject } from "rxjs";

@Injectable()

export class BlogService{

private readonly baseURL: string;

postId**Source** = new BehaviorSubject<number>(0);

postIdData: any;

constructor(private http: HttpClient){

this.baseURL ="https://localhost:44314/api/post/";

this.postIdData= this.postId**Source**.asObservable();

}

getCategoryList(){

let header = new HttpHeaders();

header.append('Content-Type', 'applications/json');

return this.http.get(this.baseURL + "getcategories", { headers: header})

}

getPostList(){

let header = new HttpHeaders();

header.append('Content-Type', 'applications/json');

return this.http.get(this.baseURL + "getposts", { headers: header})

}

addPost(post: any){

let header = new HttpHeaders();

header.append('Content-Type', 'applications/json');

return this.http.post(this.baseURL + "addpost", post, { headers: header})

}

deletePost(postId: number){

let header = new HttpHeaders();

header.append('Content-Type', 'applications/json');

return this.http.post(this.baseURL + "deletepost?postId="+postId, { headers: header})

}

updatePost(post: any){

let header = new HttpHeaders();

header.append('Content-Type', 'applications/json');

return this.http.post(this.baseURL + "updatepost", post, { headers: header})

}

getPost(postId: number){

let header = new HttpHeaders();

header.append('Content-Type', 'applications/json');

return this.http.get(this.baseURL + "getpost?postId="+ postId, { headers: header})

}

changePostId(postId: number){

this.postId**Source**.next(postId);

}

}

So, we have service ready now. It means, we can do further implementation without any break. So, let first configure the **bootstrap**and **glyphicons**inside the **Angular.json** as follows.

**angular.json**

Hide   Copy **Code**

"styles": [

"node\_modules/bootstrap/dist/css/bootstrap.css",

"src/styles.css"

],

"scripts": [

"node\_modules/glyphicons/glyphicons.js"

]

Now create a new folder name as '**components**' and jump to this folder in terminal windows and execute the following commands to create three new components.

* **ng generate component AddNewPost**
* **ng generate component EditPost**
* **ng generate component DeletePost**

Once executing these three commands, our **AppModule**will look like as follow. Here, we would like to confirm a few things.

* Import **FormsModule**, **ReactiveFormsModule** for using forms related functionality.
* Import **HttpClientModule** for accessing API and perform database **operation**s.
* Import **ModalModule, BsModalService** to create a Modal Popup.
* Import **BlogService** to access **CRUD** functions which we have already defined above.
* Add above these components inside the EntryComponents section, so that we can use these components as a Popup.

**app.module.ts**

Hide   Shrink Image 2 for CRUD Operations in Angular with Typescript using ngx-bootstrap  Copy **Code**

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

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

import { FormsModule, ReactiveFormsModule } from "@**angular**/forms";

import { HttpClientModule } from '@**angular**/common/http';

import { ModalModule, BsModalService } from 'ngx-bootstrap/modal';

import { AppRoutingModule } from './app-routing.module';

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

import { BlogService } from 'src/app/services/blog.service';

import { AddNewPostComponent } from './components/add-new-post/add-new-post.component';

import { EditPostComponent } from './components/edit-post/edit-post.component';

import { DeletePostComponent } from './components/delete-post/delete-post.component';

@NgModule({

declarations: [

AppComponent,

AddNewPostComponent,

EditPostComponent,

DeletePostComponent

],

imports: [

BrowserModule,

FormsModule,

ReactiveFormsModule,

AppRoutingModule,

HttpClientModule,

ModalModule.forRoot()

],

providers: [BlogService, BsModalService],

bootstrap: [AppComponent],

entryComponents:[AddNewPostComponent, DeletePostComponent, EditPostComponent]

})

export class AppModule { }

Let's open **app.component.html**and create the list for showing the available posts. We will create the list using table and **\*ngFor**directive. We are using **Bootstrap** CSS classes for the look and feel and **glyphicon**for Icons. This table will also contain the two button for **Editing**the post and **Deleting**the post. Apart from these, we will keep one more button at the top which will open a dialog from where we can **add a new post**.

**app.component.html**

Hide   Shrink Image 3 for CRUD Operations in Angular with Typescript using ngx-bootstrap  Copy **Code**

<router-outlet></router-outlet>

<div class="container">

<div class="row">

<div class="col-md-6" style="text-align:center;">

<h3>**CRUD** **Operation** in **Angular** with TypeScript</h3>

</div>

</div>

<div class="row">

<div class="col-md-8">

<span (click)="addNewPost()" class="btn btn-primary btn-xs">Add New Post</span>

<br /><br />

<div class="table-responsive">

<table id="mytable" class="table table-bordred table-striped">

<thead>

<th>Post Id</th>

<th>Title</th>

<th>Description</th>

<th>Category</th>

<th>Create Date</th>

<th>Edit</th>

<th>Delete</th>

</thead>

<tbody>

<tr \*ngFor="let item of postList">

<td>{{item.postId}}</td>

<td>{{item.title}}</td>

<td>{{item.description}}</td>

<td>{{item.categoryName}}</td>

<td>{{item.createdDate | date}}</td>

<td><span class="btn btn-primary btn-xs" (click)="editPost(item.postId)">

<span class="glyphicon glyphicon-pencil"></span></span></td>

<td><span class="btn btn-danger btn-xs" (click)="deletePost(item.postId, item.title)">

<span class="glyphicon glyphicon-trash"></span></span></td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

</div>

Now, we have designed our app,component.html page where we will list of data along with **Add, Update, Delete button**. But now it's time to add the actual functionality for these buttons. So, move to **AppComponent** to add the functionality to **open the Dialog** Window for each button click where we will do the actual implementation of **CRUD operations**. But here, you can see we are opening the Modal Popup on each button click.

* On the page initialization, we will get the list of available posts from the service using **getPosts()** function which will bind with 'app.component.html' table using \*ngFor directive.
* **AddNewPost()** function will open a Modal Popup from where we can add a new post, once new record will add, our list will be updated.
* **DeletePost()** function will open a Modal Popup where we will send the PostId and Title of the Post and delete the individual post.
* **EditPost()** function will open a Modal Popup where we will send the PostId using Rxjs and update that particular post.

**app.component.ts**

Hide   Shrink Image 4 for CRUD Operations in Angular with Typescript using ngx-bootstrap  Copy **Code**

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

import { BlogService } from 'src/app/services/blog.service';

import { BsModalService, BsModalRef } from "ngx-bootstrap/modal";

import { AddNewPostComponent } from './components/add-new-post/add-new-post.component';

import { DeletePostComponent } from './components/delete-post/delete-post.component';

import { EditPostComponent } from './components/edit-post/edit-post.component';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = '**AngularCRUD**Example';

postList: any[] = [];

bsModalRef: BsModalRef;

constructor(private blogService: BlogService, private bsModalService: BsModalService) {

this.getPosts();

}

getPosts() {

this.blogService.getPostList().subscribe(data => {

Object.assign(this.postList, data);

}, error => {

console.log("Error while getting posts ", error);

});

}

addNewPost() {

this.bsModalRef = this.bsModalService.show(AddNewPostComponent);

this.bsModalRef.content.event.subscribe(result => {

if (result == 'OK') {

this.getPosts();

}

});

}

deletePost(postId: number, title: string) {

this.bsModalRef = this.bsModalService.show(DeletePostComponent);

this.bsModalRef.content.postId = postId;

this.bsModalRef.content.title = title;

this.bsModalRef.content.event.subscribe(result => {

console.log("deleted", result);

if (result == 'OK') {

setTimeout(() => {

this.postList=[];

this.getPosts();

}, 5000);

}

});

}

editPost(postId: number) {

this.blogService.changePostId(postId);

this.bsModalRef = this.bsModalService.show(EditPostComponent);

this.bsModalRef.content.event.subscribe(result => {

if (result == 'OK') {

setTimeout(() => {

this.getPosts();

}, 5000);

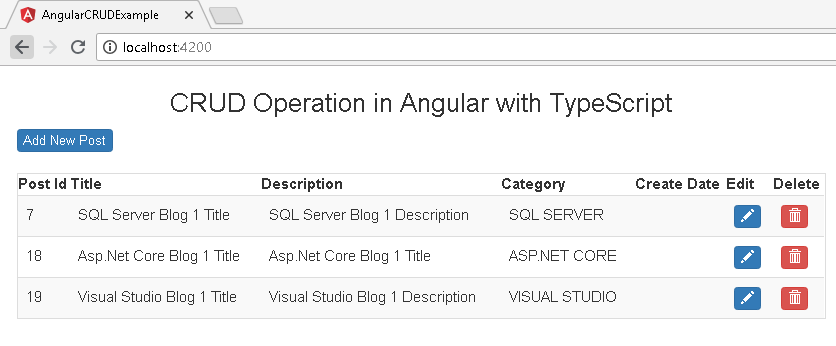
}

});

}

}

Now, let's run the app using **ng serve** command see the progress till now. Wow... it looks good and showing the list of records from the database.



Above we have done with showing the records from the database, now we will add a new record into the database. So, let's move to **add-new-post.component.html** and replace with following **code**s. Here you can see we are creating one **Reactive Forms** with required controls for adding the data along with two buttons as **Submit and Close**. We have two different types of control with this forms, one is dropdown and others are textboxes.

**add-new-post.component.html**

Hide   Shrink Image 6 for CRUD Operations in Angular with Typescript using ngx-bootstrap  Copy **Code**

<div role="document">

<div>

<form class="form-horizontal" id="add-form" [formGroup]="addNewPostForm" (ngSubmit)="onPostFormSubmit()">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-label="Close" (click)="onClose()"><span aria-hidden="true">&times;</span></button>

<h4 class="modal-title" id="add-modal-label">Add Post</h4>

</div>

<div class="modal-body">

<div class="form-group">

<label class="col-sm-2 control-label">Cateogry</label>

<div class="col-sm-10">

<select formControlName="category">

<option [value]="null">Select Category</option>

<option \*ngFor="let item of categories" [value]="item.id">{{item.name}}</option>

</select>

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label">Title</label>

<div class="col-sm-10">

<input type="text" class="form-control" formControlName="title" />

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label">Description</label>

<div class="col-sm-10">

<textarea class="form-control" formControlName="description"></textarea>

</div>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-default" data-dismiss="modal" (click)="onClose()">Close</button>

<button type="submit" class="btn btn-primary">Submit</button>

</div>

</form>

</div>

</div>

Now, let's move to **AddNewPostComponent** for the actual implementation of add logic. First, we will import required module classes and then create the Reactive Forms inside the constructor with controls' default value. Then get the list of categories, which will bind with a dropdown. There is one more method as **onPostFormSubmit()** we will get the all control's values like **Title, Description, Category Id** using the form's instance and pass it to service for adding a new record. Here we have one more function defined as **onClose()** which will close the**Modal Popup.**

**add-new-post.component.ts**

Hide   Shrink Image 7 for CRUD Operations in Angular with Typescript using ngx-bootstrap  Copy **Code**

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

import { FormBuilder, FormGroup, FormControl } from '@**angular**/forms';

import { BlogService } from 'src/app/services/blog.service';

import { BsModalRef } from 'ngx-bootstrap/modal';

@Component({

selector: 'app-add-new-post',

templateUrl: './add-new-post.component.html',

styleUrls: ['./add-new-post.component.css']

})

export class AddNewPostComponent implements OnInit {

addNewPostForm: FormGroup;

categories: any[] = [];

event: EventEmitter<any>=new EventEmitter();

constructor(private builder: FormBuilder, private blogService: BlogService, private bsModalRef: BsModalRef) {

this.addNewPostForm = this.builder.group({

category: new FormControl(null, []),

title: new FormControl('', []),

description: new FormControl('', [])

});

this.blogService.getCategoryList().subscribe(data => {

Object.assign(this.categories, data);

}, error => { console.log('Error while gettig category data.'); });

}

onPostFormSubmit(){

let postData = {

'Title': this.addNewPostForm.get('title').value,

'Description': this.addNewPostForm.get('description').value,

'CategoryId': this.addNewPostForm.get('category').value,

};

this.blogService.addPost(postData).subscribe(data=>{

console.log(data);

if(data!=null && data>0){

this.event.emit('OK');

this.bsModalRef.hide();

}

});

}

onClose(){

this.bsModalRef.hide();

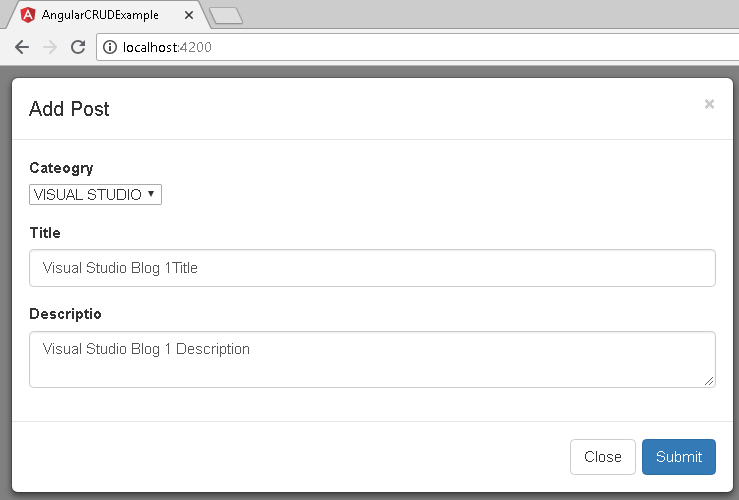
}

ngOnInit() {

}

}

Now, let's run the app again using **ng serve**command see the progress till now. This time click '**Add New Post**' button from the main page, it will open the Modal Popup similar to below image. Now select the category from the dropdown, fill the Title and Description values in textboxes and click to Submit button. It will save the value inside the database and return back to the main page with updated data.



Now its time to add Edit functionality for the post, we can edit the existing post when clicking on the**Edit**Icon in the table for a particular record. So, let's move to**edit-post.component.html**and replace with following **code**s. Here you can see we are creating one Reactive Forms for editing the existing record with required controls along with two buttons as Update and Close. Let's first create this forms and then we will implement the logic for Editing the existing record.

**edit-post.component.html**

Hide   Shrink Image 9 for CRUD Operations in Angular with Typescript using ngx-bootstrap  Copy **Code**

<div role="document">

<div>

<form class="form-horizontal" id="add-form" [formGroup]="editPostForm" (ngSubmit)="onPostEditFormSubmit()">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-label="Close" (click)="onClose()"><span aria-hidden="true">&times;</span></button>

<h4 class="modal-title" id="add-modal-label">Edit Post</h4>

</div>

<div class="modal-body">

<div class="form-group">

<label class="col-sm-2 control-label">Cateogry</label>

<div class="col-sm-10">

<select formControlName="category">

<option [value]="null">Select Category</option>

<option \*ngFor="let item of categories" [value]="item.id">{{item.name}}</option>

</select>

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label">Title</label>

<div class="col-sm-10">

<input type="text" class="form-control" formControlName="title" />

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label">Descriptio</label>

<div class="col-sm-10">

<textarea class="form-control" formControlName="description"></textarea>

</div>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-default" data-dismiss="modal" (click)="onClose()">Close</button>

<button type="submit" class="btn btn-primary">Update</button>

</div>

</form>

</div>

</div>

Now move to **EditPostComponent**and create the form with default value using Reactive Forms approach inside the constructor. And then get the list of categories and bind the drop-down. Apart from this, we have Post Id available for which we will update the data. So, first let gets the existing data for that particular Post Id and bind with control's current data.

Hide   Copy **Code**

this.editPostForm.controls['category'].setValue(this.postData.categoryId);

this.editPostForm.controls['title'].setValue(this.postData.title);

this.editPostForm.controls['description'].setValue(this.postData.description);

Once actual values will be set up then we can modify the value. After modifying the existing value and clicking on **Update**button, it will call to **onPostEditFormSubmit()** function, this function is responsible for gathering the updated values from the form and update to the database using **BlogService's**EditPost function.

**edit-post.component.ts**

Hide   Shrink Image 10 for CRUD Operations in Angular with Typescript using ngx-bootstrap  Copy **Code**

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

import { FormGroup, FormBuilder, FormControl } from '@**angular**/forms';

import { BlogService } from 'src/app/services/blog.service';

import { BsModalRef } from 'ngx-bootstrap/modal';

@Component({

selector: 'app-edit-post',

templateUrl: './edit-post.component.html',

styleUrls: ['./edit-post.component.css']

})

export class EditPostComponent implements OnInit {

editPostForm: FormGroup;

categories: any[] = [];

postId: number;

postData: any;

event: EventEmitter<any> = new EventEmitter();

constructor(private builder: FormBuilder, private blogService: BlogService, private bsModalRef: BsModalRef) {

this.editPostForm = this.builder.group({

category: new FormControl(null, []),

title: new FormControl('', []),

description: new FormControl('', [])

});

this.blogService.getCategoryList().subscribe(data => {

Object.assign(this.categories, data);

}, error => { console.log('Error while gettig category data.'); });

this.blogService.postIdData.subscribe(data => {

this.postId = data;

if (this.postId !== undefined) {

this.blogService.getPost(this.postId).subscribe(data => {

this.postData = data;

if (this.editPostForm!=null && this.postData!=null) {

this.editPostForm.controls['category'].setValue(this.postData.categoryId);

this.editPostForm.controls['title'].setValue(this.postData.title);

this.editPostForm.controls['description'].setValue(this.postData.description);

}

}, error => { console.log("Error while gettig post details") });

}

});

}

onPostEditFormSubmit() {

let postData = {

'PostId': this.postId,

'Title': this.editPostForm.get('title').value,

'Description': this.editPostForm.get('description').value,

'CategoryId': this.editPostForm.get('category').value,

};

this.blogService.updatePost(postData).subscribe(data => {

this.event.emit('OK');

this.bsModalRef.hide();

});

}

onClose() {

this.bsModalRef.hide();

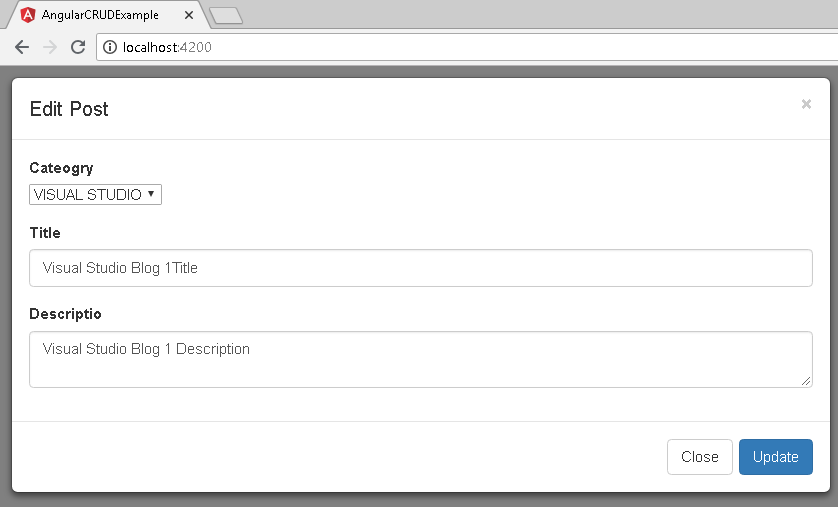
}

ngOnInit() {

}

}

Now, let us test the edit functionality. So, first run the app using **ng serve**command and from the list just click to edit icon for any particular records. It will open an edit **Modal Popup**similar to below images. So, change the actual  values with the new one and click to Update button. It will update your existing record with new data and reflect to list as well with updated data.



Till now, we have done **Add** and **Edit** functionality. Now its time to add **Delete functionality**for the post, we can delete the existing post when clicking on the Delete Icon in the table for a particular record. So, let's move to **delete-post.component.html** and replace with following **code**s. Here you can see we have only two buttons as Delete and Close. Apart from this, on the top, we will show the Title of the post for confirmation that you are deleting the right post.

**delete-post.component.html**

Hide   Copy **Code**

<div role="document">

<div>

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-label="Close" (click)="onClose()"><span aria-hidden="true">&times;</span></button>

<h4 class="modal-title" id="add-modal-label">Delete Post</h4>

</div>

<div class="modal-body">

<div class="form-group">

<label class="col-sm-2 control-label">Are you sure to delete following post?</label>

<div class="col-sm-10">

<h3 style="color:blue;">{{title}}</h3>

</div>

</div>

<div class="form-group">

<label class="col-sm-2 control-label"></label>

<div class="col-sm-10">

<button type="button" class="btn btn-default" data-dismiss="modal" (click)="onClose()">Close</button>

<button type="submit" class="btn btn-primary" (click)="deletePost(postId)" style="margin:5px;">Delete</button>

</div>

</div>

</div>

</div>

</div>

**OK**, we have designed the page for delete the existing the post. So, let's move to **DeletePostComponent** and implement the delete post functionality. On button click, we will call **deletePost()** function from the BlogService which takes Post Id as a parameter. Once your post will delete, Modal Popup will hide and your list will update.

**delete-post.component.ts**

Hide   Shrink Image 12 for CRUD Operations in Angular with Typescript using ngx-bootstrap  Copy **Code**

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

import { BsModalRef } from 'ngx-bootstrap/modal';

import { BlogService } from 'src/app/services/blog.service';

@Component({

selector: 'app-delete-post',

templateUrl: './delete-post.component.html',

styleUrls: ['./delete-post.component.css']

})

export class DeletePostComponent implements OnInit {

postId: number;

title: string;

event: EventEmitter<any> = new EventEmitter();

constructor(private bsModalRef: BsModalRef, private blogService: BlogService) {

}

deletePost(postId: number) {

this.blogService.deletePost(postId).subscribe();

this.event.emit('OK');

this.bsModalRef.hide();

}

onClose() {

this.bsModalRef.hide();

}

ngOnInit() {

}

}

So, finally let's test the delete functionality. First, run the project using**ng serve** command and click to **Delete**icon from the list of post. It will open the Modal Popup similar to below screen. Once you will click to Delete button, it will delete the record and return back to the main page where you can see a list of post.

