Angular 8

Contents

[Create Singe Entry Form Page 2](#_Toc11317403)

[Create component controller 2](#_Toc11317404)

[Create component html view 2](#_Toc11317405)

[Register component and define routes 3](#_Toc11317406)

[Create List Page with pagination and search 4](#_Toc11317407)

[Create component controller 4](#_Toc11317408)

[Create component list html view 4](#_Toc11317409)

[Common Components 5](#_Toc11317410)

# Create Singe Entry Form Page

Single entry form add or edit a single record. For example, add/edit College, Student or Marksheet

## Create component controller

1. student.component.ts file
2. Define StudentComponent class that inherits BaseCtl class in this file. Make sure selector and templateUrl properties have correct entry.

@Component({

selector: 'app-student',

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

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

})

export class StudentComponent extends BaseCtl { .. }

1. Define constructor in this class and inject ServiceLocatorService and ActivatedRoute object.

constructor(public locator: ServiceLocatorService,

public route: ActivatedRoute) {

super(locator.endpoints.STUDENT, locator, route);

}

1. Make sure pass respective server endpoint to super constructor. Here end point is locator.endpoints.STUDENT.
2. Override validateForm(form) method to validate input html form elements. Add properties which will be validated.

validateForm(form) {

let flag = true;

let validator = this.serviceLocator.dataValidator;

flag = flag && validator.isNotNullObject(form.firstName);

flag = flag && validator.isNotNullObject(form.lastName);

flag = flag && validator.isNotNullObject(form.mobileNo);

return flag;

}

1. Override populateForm(form, data) method to populate data, fetched from into input form. Make entries of properties received form server.

populateForm(form, data) {

form.id = data.id;

form.collegeId = data.collegeId;

form.email = data.email;

form.firstName = data.firstName;

form.lastName = data.lastName;

form.mobileNo = data.mobileNo;

}

## Create component html view

1. student.component.html file
2. Define placeholder to display error and success message of input form:

<div \*ngIf="form.error" style="color:red"> {{form.message}}</div>

<div \*ngIf="!form.error" style="color:green">{{form.message}}</div>

1. Define html form to enter user data and create placeholders to display input error messages. Form data are bound with form.data object and input error messages are bund with form.inouterror object.

<form>

FirstName:<input name="fName" [(ngModel)]="form.data.firstName">

<span style="color: red">{{form.inputerror.firstName}}</span>

Last Name:<input name="lName" [(ngModel)]="form.data.lastName">

<span style="color: red"> {{form.inputerror.lastName}} </span>

Email: <input name="email" [(ngModel)]="form.data.email">

<span style="color: red"> {{form.inputerror.email}} </span>

<button (click)="submit()" [disabled]="!validate()">Save</button>

<button (click)="forward('/studentlist')" >List</button>

</form>

1. Save button will call “submit()” method to submit data to the server.

<button (click)="submit()" [disabled]="!validate()">Save</button>

1. If you want to navigate to some other page on button click then call forward(/url) method. Here we are forwarding request to student list page forward('/studentlist').

## Register component and define routes

1. Register component in app.module.js file

@NgModule({

declarations: [

StudentComponent,

StudentListComponent,

], .. })

1. Define routes in app-routing.module.ts file

{path: 'student', component: StudentComponent },

{path: 'student/:id', component: StudentComponent},

{path: 'studentlist', component: StudentListComponent }

1. Define link to navigate on these pages using

<a [routerLink]="['/student']">Student</a> |

<a [routerLink]="['/studentlist']">Student</a>

1. Forward to these components from any other component using forward() method:

<button (click)="forward('/studentlist')" >List</button>

# Create List Page with pagination and search

List page displayed list of records and provides pagination and searching capability.

## Create component controller

1. student-list.component.ts file
2. Define StudentListComponent class which inherits BaseListCtl class in this file

@Component({

selector: 'app-student-list',

templateUrl: './student-list.component.html',

})

export class StudentListComponent extends BaseListCtl {..}

1. Define constructor in this class and inject ServiceLocatorService and ActivatedRoute object.

constructor(public locator: ServiceLocatorService,

public route: ActivatedRoute) {

super(locator.endpoints.STUDENT, locator, route);

}

}

1. Make sure pass respective server endpoint to super constructor. Here end point is locator.endpoints.STUDENT.

## Create component list html view

List page displays list with pagination and search elements:

1. Create student-list.component.html file
2. Define placeholder to display error and success message of input form:

<div \*ngIf="form.error" style="color:red"> {{form.message}}</div>

<div \*ngIf="!form.error" style="color:green">{{form.message}}</div>

1. Define html search form to enter search data. Form data are bound with form.searchParams object.

<form>

FirstName:

<input name="fName" [(ngModel)]="form.searchParams.firstName">

Last Name:

<input name="lName" [(ngModel)]="form.searchParams.lastName">

Email:<input name="email" [(ngModel)]="form.searchParams.email">

<button (click)="submit()"> Search</button>

</form>

1. Search button will call “submit()” method to submit search data to the server.

<button (click)="submit()"> Search</button>

1. Create a table and display list elements from form.list object using \*ngFor directive.

<table>

<tr>

<th>ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>Mobile</th>

<th>Email</th>

</tr>

<tr \*ngFor="let m of form.list ">

<td> {{ m.id }} </td>

<td> {{ m.firstName }} </td>

<td> {{ m.lastName }} </td>

<td> {{ m.mobileNo }} </td>

<td> {{ m.email }} </td>

<td> <button (click)="forward( '/student/' + m.id)">Edit</button>

</td>

<td> <button (click)="delete(m.id)">Delete</button> </td>

</tr>

</table>

1. Add edit and delete button and call forward('/student/' + m.id)and delete(m.id) method to edit and delete list record.
2. Add two pagination buttons Next and Previous which will call next() and previous() methods respectively.

<button (click)="previous()" [disabled]="form.pageNo == 0" > Previous </button>

<button (click)="next()"> Next </button>

## Register component and define routes

1. Register component in app.module.js file

@NgModule({

declarations: [

StudentComponent,

StudentListComponent,

], .. })

1. Define routes in app-routing.module.ts file

{path: 'student', component: StudentComponent },

{path: 'student/:id', component: StudentComponent},

{path: 'studentlist', component: StudentListComponent }

1. Define link to navigate on these pages using

<a [routerLink]="['/student']">Student</a> |

<a [routerLink]="['/studentlist']">Student</a>

1. Forward to these components from any other component using forward() method:

<button (click)="forward('/studentlist')" >List</button>

# Common Components

Single entry form add or edit a single record. For example, add/edit College, Student or Marksheet

## HttpService

TODO

## ServiceLocatorService

TODO

## EndpointService

TODO

## DataValidator

TODO

## BaseCtl

TODO

## BaseListCtl

TODO

# In build Angular components

## ActivatedRoute

TODO

## Router

TODO

## HttpClient

TODO