**TEMPLATE DRIVEN FORM**

**Question 1**

**Template Driven Form - Create a new component for editing employee details**  
  
We are about to gradually build a form to edit employee details with validations  
  
As a first step, we will create a new component for this and include a form. Complete the steps specified below:

1. Create a new component with name "edit-emp-template-driven" in "angular-learning" app
2. Include new section in header with title "Edit Employee (Template Driven)"
3. Link component with header using routing
4. In component html Include a form with textbox. Name the textbox as "name".
5. Below is the output expected

|  |
| --- |
| https://cognizant.e-box.co.in/uploads/Image/01fseangular/tdf2.png |

**Answer 1**

*All other files same as previous.*

**Employee.ts**

import { Department } from './Department'

import { Skill } from './Skill';

export class **Employee**

{

    id:number;

    name:string;

   salary:number;

    permanent :boolean;

    department:**Department**;

    skills:**Skill**[];

    dateofBirth:**Date**;

}

**Department.ts**

export class **Department**

{

    id:number;

    name:string;

}

**Skill.ts**

export class **Skill**

{

    id:number;

    name:string;

}

**app.component.html**

<b>Angular Learning</b>&nbsp;&nbsp;&nbsp;<a *routerLink*="view">View Employee</a>

&nbsp;&nbsp;<a *routerLink*="edit">Edit Employee</a>

&nbsp;&nbsp;<a *routerLink*="edit-emp-temp">Edit Employee (Template Driven)</a>

&nbsp;&nbsp;<a *routerLink*="quant-incr">Quantity Selector</a>

&nbsp;&nbsp;<a *routerLink*="quant-sel">Quantity Selector</a>

<hr>

<router-outlet></router-outlet>

**app-routing.module.ts**

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

import { Routes, RouterModule } from '@angular/router';

import { QuantityIncrementComponent } from './quantity-increment/quantity-increment.component';

import { QuantitySelectorComponent } from './quantity-selector/quantity-selector.component';

import { EditEmpComponent } from './edit-emp/edit-emp.component';

import { ViewEmpComponent } from './view-emp/view-emp.component';

import { EditEmpTemplateDrivenComponent } from './edit-emp-template-driven/edit-emp-template-driven.component';

const routes: **Routes** = [

{

path:'view',

component:ViewEmpComponent

},

{

path:'edit',

component:EditEmpComponent

},

{

path:'quant-incr',

component:QuantityIncrementComponent

},

{

path:'quant-sel',

component:QuantitySelectorComponent

},

{

path:'edit-emp-temp',

component:EditEmpTemplateDrivenComponent

}

];

@**NgModule**({

imports: [RouterModule.**forRoot**(routes)],

exports: [RouterModule]

})

export class **AppRoutingModule** { }

**edit-emp-template-driven.component.ts**

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

import { Employee } from '../Employee';

import { Department } from '../Department';

@**Component**({

selector: 'app-edit-emp-template-driven',

templateUrl: './edit-emp-template-driven.component.html',

styleUrls: ['./edit-emp-template-driven.component.css']

})

export class **EditEmpTemplateDrivenComponent** implements **OnInit** {

constructor() { }

**ngOnInit**() {

}

  departments: **Department**[]=[

    {id:1,name:"HelpDesk"},

    {id:2,name:"HR"},

    {id:3,name:"IT"},

    {id:4,name:"Payroll"},

    {id:5,name:"admin"}

  ];

  emp:**Employee**={

    id:null,

    name:null,

    salary:null,

    permanent:null,

    department:{

      id:1,

      name:'Payroll'

    },

    skills:[

      {

        id:1,

        name:'HTML'

      },

      {

        id:2,

        name:'CSS'

      },

      {

        id:3,

        name:'Javascript'

      }

    ],

    dateofBirth: new **Date**('04/20/2019')

}

**saveForm**(newEmployee:**Employee**):void{

**console**.**log**(newEmployee);

  }

}

**edit-emp-template-driven.component.html**

<div *style*="float: left;width: 33%;">

<h2>

    Edit Employee

</h2>

<label *for*="name">Name</label>

<br/>

<input *type*="text" *name*="name" *id*="name"/>

</div>

**Question 2**

**Template Driven Form - Template Reference Variable and binding with NgModel**  
  
Include a Template Reference Variable for name textbox and display the value. Refer code below.

<input type="text" name="name" #name>

{{name.value}}

The above code will not display the value when typed in a textbox, since the field is not bound to model data in component.  
  
Perform below steps to do the ngModel binding:

1. Import FormModule in app.module.ts
2. Include FormModule in imports of app.module.ts
3. Include employee property in component
4. Assign static data to this property by copying and pasting the code from view-emp component
5. update the input textbox in html as specified below

<input type="text" name="name" [(ngModel)]="employee.name" #name="ngModel">

1. Try changing the value in name field which will be reflected immediately outside the textbox.

Sample output below

|  |
| --- |
| https://cognizant.e-box.co.in/uploads/Image/01fseangular/tdf3(1).png |

**Answer 2**

**edit-emp-template-driven.component.html**

<div *style*="float: left;width: 33%;">

<h2>

    Edit Employee

</h2>

<label *for*="name">Name</label>

<br/>

<input *type*="text" *name*="name" *id*="name" *[(ngModel)]*="emp.name" *#empName*="ngModel"/>

{{emp.name}}

</div>

**Question 3**

**Template Driven Form - Validation properties of ngModel**  
  
As data was bound in the previous hands on, let us understand the validation related properties available  as part of this binding.  
  
Perform instructions as specified below:

* Include required, minlength and maxlength attributes to the name text field.
* Set minlength as 2 and maxlength as 10
* Include display of various properties related to validation. Refer code below:

<div>name.invalid - {{name.invalid}}</div>

<div>name.touched - {{name.touched}}</div>

<div>name.dirty - {{name.dirty}}</div>

<div>name.errors.required - {{name.errors?.required}}</div>

<div>name.errors.minlength - {{name.errors?.minlength | json}}</div>

NOTE: ? denotes to ignore at runtime if undefined

* Try changing the value in the textbox in below specified aspects and observe how the validation properties change:
  + Clear all text
  + Enter text less than 2 characters
  + Observe when touched, dirty, invalid and errors values change

SME to walkthrough on all scenarios when each property will have true or false.  
  
Sample Layout

|  |
| --- |
| https://cognizant.e-box.co.in/uploads/Image/01fseangular/tdf4.JPG |

**Answer 3**

**edit-emp-template-driven.component.html**

<div *style*="float: left;width: 33%;">

<h2>

    Edit Employee

</h2>

<label *for*="name">Name</label>

<br/>

<input *type*="text" *name*="name" *id*="name" *[(ngModel)]*="emp.name" *#empName*="ngModel"/>

<div>name.invalid - {{empName.invalid}}</div>

<div>name.touched - {{empName.touched}}</div>

<div>name.dirty - {{empName.dirty}}</div>

<div>name.errors.required - {{empName.errors?.required}}</div>

<div>name.errors.minlength - {{empName.errors?.minlength | json}}</div>

</div>

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

**Question 4**

**Template Driven Form - Validation for name field**  
  
Implement validation for name field on the below specified criteria:

* Name field is required
* Name must at least have 2 characters
* Name can have a maximum of 10 characters

Based on the validation error respective message needs to be displayed below the text box in smaller font red color. For red font color, include style in component.css file.  
  
**Sample Layout**

|  |
| --- |
| https://cognizant.e-box.co.in/uploads/Image/01fseangular/tdf5.PNG |

**Answer 4**

**edit-emp-template-driven.component.html**

<div *style*="float: left;width: 33%;">

<h2>

    Edit Employee

</h2>

<label *for*="name">Name</label>

<br/>

<input *type*="text" *name*="name" *id*="name" *[(ngModel)]*="emp.name" *#empName*="ngModel"/>

<span *class*="help-block" *\*ngIf*="empName.errors?.required && empName.touched">

Name is Required

</span>

<span *class*="help-block" *\*ngIf*="empName.errors?.minlength">

Name Must be at least 2 characters long

</span>

<span *class*="help-block" *\*ngIf*="empName.errors?.maxlength">

Name can have maximum of 10 characters

</span>

</div>

**Question 5**

**Template Driven Form - Include salary and permanent field**  
  
Before including additional fields, let us include more details in the screen about the data available in the background. Let us display the employee data and form data. Follow steps below:

* Include below block style in component css file

.block {

float:left;

width:33%

}

* Enclose each columnar section (refer screenlayout below) with div having block as style.
* Include below code in the end for second column with title "Employee Model"

<div class="block">

<strong>Employee Model</strong><br>

<pre>{{employee | json }}</pre>

</div>

* Define template reference variable for the form as #empForm
* Include below code in the end for third column with title "Angular Generated Form Model"

<div class="block">

<strong>Angular Generated Form Model</strong><br>

<pre>{{empForm.value | json}}</pre>

</div>

Incorporate the below aspects of the edit employee screen:

* Include salary and mandatory field validation.
* Include permanent field using radio buttons.
* Important points to implement for 'permanent' radio buttons:
  + Define both radio buttons with same name 'permanent'.
  + Define each button with different id as permanentTrue and permanentFalse respectively.
  + Define value in square brackets (one way binding) with value as "true" and "false" respectively.
  + Bind [(ngModel)] to employee.permanent.

**Sample Screen Layout**

|  |
| --- |
| https://cognizant.e-box.co.in/uploads/Image/01fseangular/tdf7.png |

**Answer 5**

**edit-emp-template-driven.component.html**

<div *style*="float: left;width: 33%;">

<h2>

    Edit Employee

</h2>

<form >

<div *class*="form-group">

<label *for*="ID">ID</label>

<br/>

<input *required* *type*="number" *name*="ID" *id*="ID" *[(ngModel)]*="emp.id" *#empID*="ngModel"/>

</div>

<div *class*="form-group">

<label *for*="name">Name</label>

<br/>

<input *type*="text" *name*="name" *id*="name" *[(ngModel)]*="emp.name" *#empName*="ngModel"/>

<span *class*="help-block" *\*ngIf*="empName.errors?.required && empName.touched">

Name is Required

</span>

<span *class*="help-block" *\*ngIf*="empName.errors?.minlength || empName.errors?.maxlength">

Name Must be at least 2 charcters long

</span>

</div>

<div *class*="form-group">

<label *for*="salary">Salary</label>

<br/>

<input *type*="number" *name*="salary" *id*="salary" *[(ngModel)]*="emp.salary" *#empSalary*="ngModel" />

</div>

<div *class*="form-group">

<div>

<label *for*="control-label">Is Permanent or not ?</label>

<br/>

<label *class*="radio-inline">

<input *type*="radio" *name*="permanent" *id*="permanent" *value*="true" *[(ngModel)]*="emp.permanent" *#empPermananent*="ngModel" /> Yes

</label>

<label *class*="radio-inline">

<input *type*="radio" *name*="permanent" *id*="permanent" *val*="false" *[(ngModel)]*="emp.permanent" *#empPermananent*="ngModel" /> No

</label>

</div>

</div>

</form>

</div>

<div *style*="float:left;width: 33%" >

<strong>Employee Model</strong>

<br/>

<pre>{{emp|json}}</pre>

</div>

<div *style*="float: left;width:33%">

<strong>Angular Genrated Form Model</strong>

<br/>

<pre>{{empform.value|json}}</pre>

</div>

**Question 6**

**Template Driven Form - Include department drop down and form submission**  
  
The department drop down should be populated based on a list. Respective option in the drop down should be selected based on the department data in employee.department property. Form submission needs to be done by incorporating a submit button.  
  
Implement below steps to incorporate department drop down and submit the form:

* Include departments property in component.ts and add departments. This will be used for the items in the department dropdown. Refer code below

this.departments = [

{ id: 1, name: "Payroll" },

{ id: 2, name: "Internal" },

{ id: 3, name: "HR" }

];

* Define <select> with ngModel as employee.department.id
* Apply \*ngFor for <option> with below attributes
  + Map [value] with 'department.id' with one way binding.
  + Map [selected] with condition 'employee.department.id == department.id'. This make the respective drop down item selected
  + Include <option> tag with department.name as interpolation.
  + Include (change) event in <select> with assignment of department id using employee.department.id = $event.target.value.
* Include form submission as specified below. This ensures submit button is enabled only after all the validations are successful.

<button type="submit" [disabled]="!empForm.valid">Save</button>

* Bind form submission to a function in component by addting attribute (ngSubmit)="onSubmit()" to form. Include relevant function in component to log the employee details in console.

**Sample Layout**

|  |
| --- |
| https://cognizant.e-box.co.in/uploads/Image/01fseangular/tdf6.png |

**Answer 6**

**edit-emp-template-driven.component.html**

<div *style*="float: left;width: 33%;">

<h2>

    Edit Employee

</h2>

<form *(ngSubmit)*="saveForm(employee)" *#empform*="ngForm">

<div *class*="form-group">

<label *for*="ID">ID</label>

<br/>

<input *required* *type*="number" *name*="ID" *id*="ID" *[(ngModel)]*="emp.id" *#empID*="ngModel"/>

</div>

<div *class*="form-group">

<label *for*="name">Name</label>

<br/>

<input *type*="text" *name*="name" *id*="name" *[(ngModel)]*="emp.name" *#empName*="ngModel"/>

<span *class*="help-block" *\*ngIf*="empName.errors?.required && empName.touched">

Name is Required

</span>

<span *class*="help-block" *\*ngIf*="empName.errors?.minlength || empName.errors?.maxlength">

Name Must be at least 2 charcters long

</span>

</div>

<div *class*="form-group">

<label *for*="salary">Salary</label>

<br/>

<input *type*="number" *name*="salary" *id*="salary" *[(ngModel)]*="emp.salary" *#empSalary*="ngModel" />

</div>

<div *class*="form-group">

<div>

<label *for*="control-label">Is Permanent or not ?</label>

<br/>

<label *class*="radio-inline">

<input *type*="radio" *name*="permanent" *id*="permanent" *value*="true" *[(ngModel)]*="emp.permanent" *#empPermananent*="ngModel" /> Yes

</label>

<label *class*="radio-inline">

<input *type*="radio" *name*="permanent" *id*="permanent" *val*="false" *[(ngModel)]*="emp.permanent" *#empPermananent*="ngModel" /> No

</label>

</div>

</div>

<div>

<label *for*="department">Department : </label>

<br/>

<select *id*="department" *name*="department" *[(ngModel)]*="emp.department">

<option *\*ngFor*="let dept of departments" *[value]*="dept.id">

{{dept.name}}

</option>

</select>

</div>

<br/>

<div *style*="margin-left:150px;margin-bottom: 20px;">

<button *type*="submit" *class*="btn" *[disabled]*="empform.invalid">Save</button>

</div>

</form>

</div>

<div *style*="float:left;width: 33%" >

<strong>Employee Model</strong>

<br/>

<pre>{{emp|json}}</pre>

</div>

<div *style*="float: left;width:33%">

<strong>Angular Genrated Form Model</strong>

<br/>

<pre>{{empform.value|json}}</pre>

</div>