**Lesson12 Events in Angular**

**Notes:-**

**1-The following bindings that we have discussed so far in this video series flow data in one direction from a component class property to an HTML element property.  
A. Interpolation  
B. Property Binding  
C. Attribute Binding  
D. Class Binding  
E. Style Binding**

**2-How about flowing data in the opposite direction from an HTML element to a component. When a user performs any action like clicking on a button, hovering over an element, selecting from a dropdown list, typing in a textbox etc, then the corresponding event for that action is raised. We need to know when user performs these actions. We can use angular event binding to get notified when these events occur**.

**Example Of using Events on Buttons**

public ButtonText:string;

public ButtonState:boolean;

ngOnInit() {

this.ButtonState = true;

this.ButtonText = "Button is Enabled";}

ClickEvent(ButtonState:boolean){

if(ButtonState){

this.ButtonState = false;

this.ButtonText = "The Button Is Disabled";}

else{

this.ButtonState = true;

this.ButtonText = "The Button is Enabled";

}}

**On The View we write the following code**

<button [disabled]="!ButtonState" (click)="ClickEvent(ButtonState)">{{ButtonText}}</button>

**Example:-**

**1-on the Employee Class , we type the following code**

export class Employee

{

ID:number;

Fname:string;

Lname:string;

Gender:string;

Salary:number;

}

**2-on the service , we type the following code**

import { HttpHeaders } from '@angular/common/http';

import { headersToString } from 'selenium-webdriver/http';

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

import { Http, Response, Headers, RequestOptions, RequestMethod } from '@angular/http';

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

//we put th e Http in the emp-service.service.ts file as the below code

import {HttpClient,HttpErrorResponse} from '@angular/common/http';

import 'rxjs/add/operator/catch';

import 'rxjs/add/observable/throw';

//we put observable in the service.ts

import { Observable } from 'rxjs/Observable';

declare var toastr:any;

@Injectable({

providedIn: 'root'

})

export class EmployeeService {

constructor(private \_http:HttpClient) { }

Employees:Employee[];

selectedEmployee:Employee;

weburl:string = 'http://localhost/WebAPIProject/api/Employee/';

GetPersons() : Observable<Employee[]>

{

debugger

const httpOptions =

{

headers: new HttpHeaders({

'accept': 'application/json'

})

};

return this.\_http.get<Employee[]>(this.weburl+"LoadEmployees",httpOptions).catch(this.errorHandler);

}

GetPerson(id:number) : Observable<Employee>

{

const httpOptions =

{

headers: new HttpHeaders({

'accept': 'application/json'

})

};

return this.\_http.get<Employee>(this.weburl+"LoadEmployee/"+id,httpOptions).catch(this.errorHandler);

}

postEmployee(emp : Employee)

{

debugger;

var body = JSON.stringify(emp);

const httpOptions =

{

headers: new HttpHeaders({

'accept': 'application/json',

'Content-Type': 'application/json'

})

};

return this.\_http.post<Employee>(this.weburl,body,httpOptions).catch(this.errorHandler);

}

putEmployee(id, emp) {

debugger;

var body = JSON.stringify(emp);

const httpOptions =

{

headers: new HttpHeaders({

'Content-Type': 'application/json',

'Authorization': 'my-auth-token'

})

};

return this.\_http.put<Employee>(this.weburl+"/"+id,body,httpOptions).catch(this.errorHandler);

}

DeletePerson(id) {

const httpOptions =

{

headers: new HttpHeaders({

'Content-Type': 'application/json',

'Authorization': 'my-auth-token'

})

};

return this.\_http.delete<Employee>(this.weburl+"/"+id,httpOptions).catch(this.errorHandler);

}

errorHandler(error:HttpErrorResponse)

{

debugger;

return Observable.throw(error.message || "Server Error");

}

}

**3-you can show hide specific columns and make it’s dynamic we type the following code**

<table>

<thead>

<tr>

<th attr.colspan="{{columnSpan}}">

Employee Details

</th>

</th><trh>

<th style="text-align: left;">

First Name

</th><th style="text-align: left;">

Last Name

//we see that we apply the show / hide the column by this expression

</th><th \*ngIf="ShowHide" style="text-align: left;">

Gender

//we see that we apply the show / hide the column by this expression

</th><th \*ngIf="ShowHide" style="text-align: left;">

Salary

</th></tr>

</thead>

<tbody>

<tr \*ngFor="let item of employeeService.Employees;">

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

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

//we see that we apply the show / hide the column by this expression

<td \*ngIf="ShowHide" >{{item.Gender}}</td>

//we see that we apply the show / hide the column by this expression

<td \*ngIf="ShowHide">{{item.Age}}</td>

</tr>

</tbody>

</table>

<button (click)="ShowHideEvent(ButtonState)">{{ShowHide ? 'Hide' : 'Show'}} Button</button>

**4-on the code behind Component we type the following code**

export class EmployeeComponent implements OnInit {

constructor(public employeeService:EmployeeService) { }

public columnSpan:number;

public ShowHide:boolean;

public lst:Employee[];

ngOnInit() {

this.ShowAllPersons();

this.ShowHide = false;

this.columnSpan = 4;

}

ShowAllPersons()

{

debugger;

this.employeeService.GetPersons().subscribe(data =>

this.employeeService.Employees = data as Employee[]

);

}

//we see that we visible or hide the columns

ShowHideEvent()

{

this.ShowHide = !this.ShowHide;

}