ANGULAR 10

Presented by

Eng./Abanoub Nabil

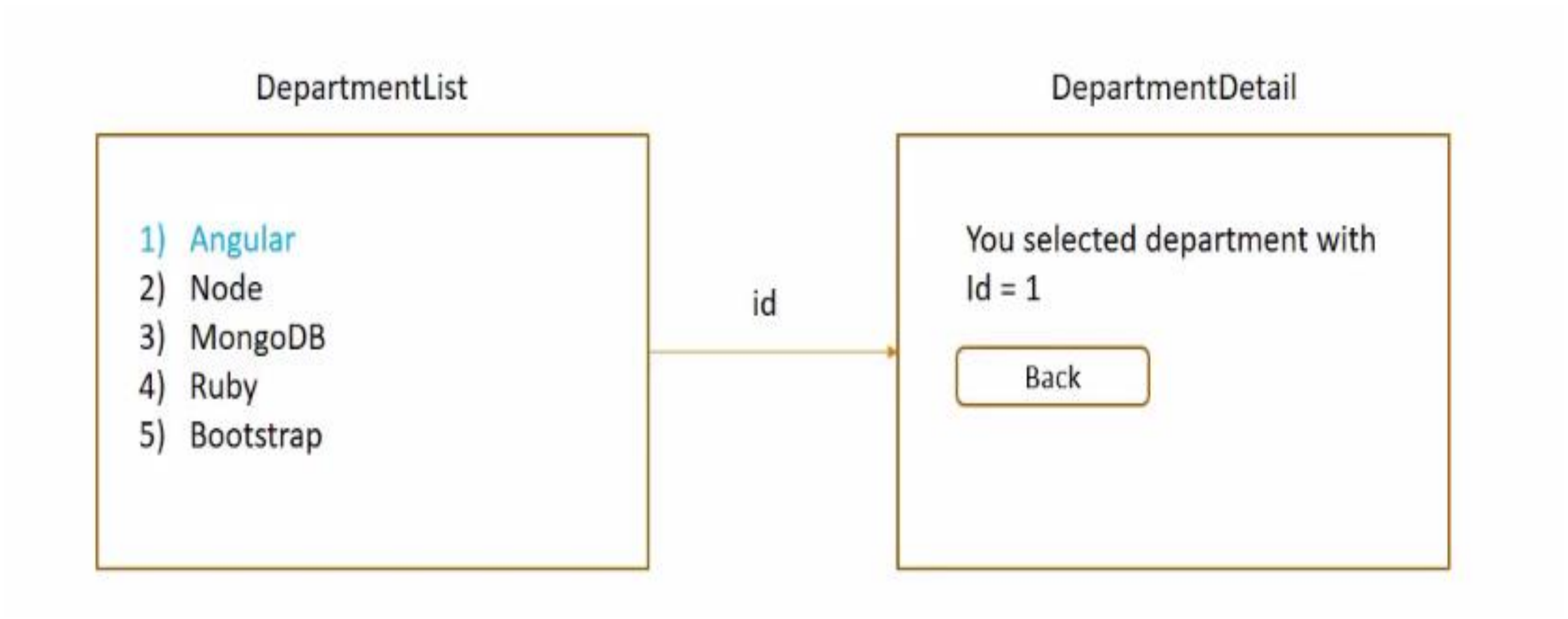
Teaching assistant at ITI

Agenda

- Optional Route Parameters
- Child Routes
- Angular Forms
- Template Driven Forms (TDF)

Optional Route Parameters

- Lets see this figure:



Optional Route Parameters(cont.)

- Here is the new requirement in the **DepartmentDetails** component we need to add back **button** when we click on the back button we should navigate back to the **DepartmentList** component and the previously selected department must be **highlighted**.

- This can be achieved by **optional route parameter.**

Optional Route Parameters(cont.)

- **1-**In the **DepartmentDetails** component lets add the back button.

```
<div>
      <button> Back </button>
</div>
```

2-We need to listen to the click event on this button

```
<div>
   <button (click)="goToDepartments()"> Back </button>
</div>
```

Optional Route Parameters(cont.)

- Lets define this method:

```
goToDepartments()
{
  this.router.navigate(['/Departments',{id:this.DeptId}])
}
```

- Now if we test we will see the id is in the URL as optional route parameter.
- **It is optional because** its existence does not affect on the view but it can be used to apply some logic to the view.

Optional Route Parameters(cont.)

- **Now we are going to read** the id from url and compare it with the all id in the list if it match one this one will be highlighted.
- Lets go the **departmentList** component and read this parameter.

Optional Route Parameters(cont.)

- **1-**We need to import activatedRoute service and inject it.

```
import {ActivatedRoute} from '@angular/router';
constructor(private route:ActivatedRoute)
```

- **2-**Then in the onInit method we are going to retrieve the parameter using paramMap Observable.

Optional Route Parameters(cont.)

```
ngOnInit() {
    this.route.paramMap.subscribe((params:ParamMap)=>{
    this.selectedDeptId=parseInt(params.get('id'));
});
}
```

- Don’t forget to import paramMap.

```
import {ParamMap} from '@angular/router';
```

Optional Route Parameters(cont.)

- 3-Now we have the id so we will compare it
  with all list id if it is match.

```
<div (click)="OnSelect(dept)" *ngFor="let dept of departments "
[class.selected]="isSelected(dept)">
    <h3>{{dept.id}} {{" "+dept.name}}</h3>
</div>

isSelected(department)
{
return department.id ===this.selectedDeptId;
}
```

Then you want to give style to the selected class.

Child Routes

- In angular application may be some routes only viewed in other routes in such a case we create child routes.
- Lets understand by example

Child Routes (cont.)

- Till now in our application we have departmentDetails component in this component lets say we want to **display an overview and the contact information of the selected department**
- The overview and contact info are features in each department so should only be viewed when navigate to this departmentDetails component

Child Routes (cont.)

- 1-Generate the two new components:

```
ng g c DepartmentOverview -it –is
ng g c DepartmentContactInfo –it -is
```

Child Routes (cont.)

- 2-Open the app-routing.module.ts to configure the new routes

```
{path :'Departments/:id',
component:DepartmentDetailsComponent,
      children:[
      {path:'overview',component:DepartmentOverviewComponent},
      {path:'contact',component:DepartmentContactInfoComponent}
      ]
}
```

Child Routes (cont.)

- We mentioned that these two components only will be displayed inside the DepartmentDetails component
- How to display this ?

  By using **router-outlet**

```
<router-outlet></router-outlet>
```

Now users can navigate to these components by writing in the URL but lets add them buttons

Child Routes (cont.)

```
<button (click)="GoToView()">Go to overview</button>
<button (click)="GoToContact()">Go to contact </button>
```

```
GoToView()
{
  this.router.navigate(['overview'],{relativeTo:this.route})
}
GoToContact()
{
  this.router.navigate(['contact'],{relativeTo:this.route})
}
```

Child Routes (cont.)

- Note: we used relative URL not absolute.

Angular Forms

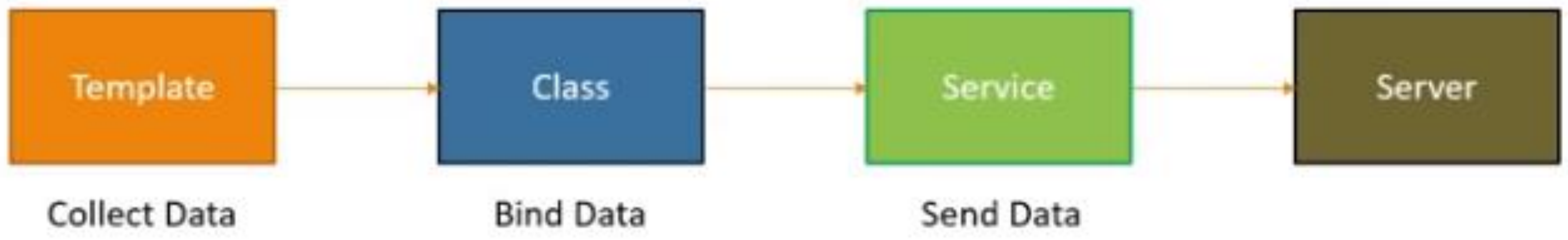
- Forms is a part of a business application.
- Like :Register , Login , Submit a request , place an order and so on…

Prerequisites

- HTML
- CSS
- JavaScript
- Angular –Templates , Components ,Data Binding and Services

Angular Forms

- The component template contains HTML to collect the user data.
- The component class handles data binding
- The collected date sent to the server through a service
-



Two approaches

- **Template Driven Forms**

  heavy on the component template
- **Reactive forms (Model Driven Forms)**

  Heavy on component class

Template Driven Forms (TDF)

- Much of the code reside in the HTML template.
- Easy to use
- Two way binding with **ngModel**
- Automatically tracks the form and form elements state and validity.
- Unit testing is a challenge.

When use TDF?

- Suitable for simple scenarios such as login and register.
- When unit testing is not necessary.

TDF (Cont.) What we will do?

- Generate a new CLI project.
- Add the form HTML.
- Biding data.
- Tracking state and validity.
- Providing visual feedback.
- Displaying error messages.
- Posting data to the server.

1-Generate a new CLI project.

- To generate a project and its features we will use angular CLI **v6 or higher**.
- To install the latest version of angular CLI use the command

```
npm install –g @angular/cli@latest
```

- To update angular CLI perform the following steps

```
npm uninstall –g @angular/cli
npm cache verify
npm install –g @angular/cli@latest
```

Bootstrap in angular application

- To be able to use bootstrap just add this in the head tag of index.html

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/b
ootstrap.min.css" integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN
9MuhOf23Q9Ifjh" crossorigin="anonymous">
```

2-Add the form HTML.

- We will make enrollment form

```
<div class="container-fluid">
      <form>
      </form>
</div>
```

- This will contain all form elements

2-Add the form HTML(cont.)

```
<div class="form-group">
  <label>Name :</label>
  <input type="text" class="form-control">
</div>
```

This add new form element of type text

2-Add the form HTML(cont.)

```
<div class="form-group">
   <label>Email :</label>
   <input type="email" class="form-control">
</div>
```

This will add new form element of type email

2-Add the form HTML(cont.)

```
<div class="form-group">
   <label>telephone :</label>
   <input type="tel"  class="form-control">
</div>
```

This will add new form element of type telephone

2-Add the form HTML(cont.)

```
<div class="form-group">
 <select class="custom-select">
          <option selected>
               I'm Interested In
          </option>

          <option *ngFor="let topic of
             topics">{{topic}}
       </option>
 </select>
</div>
```

This will add form dropdown list and bind data to it from component class.

2-Add the form HTML(cont.)

```
<div class="mb-3">
   <label>Time selection</label>
   <div class="form-check">
       <input type="radio" value="eveninig" class="form-
           check-input" name="timeSelection">
     <label class="form-check-label">Eveninig (5PM –
        8PM)</label>
</div>

<div class="form-check">
      <input type="radio" value="Morning" class="form-check-
         input" name="timeSelection">
     <label class="form-check-label">Morning (5PM –
       8PM)</label>
</div>
</div>
```

This will add form radio buttons

2-Add the form HTML(cont.)

```
<div class="form-check mb-3">
  <input type="checkbox" class="form-check-
    input">
  <label class="form-check-label">
    Send me new offers
  </label>
</div>
```

This add new checkbox element to the form

2-Add the form HTML(cont.)

```
<input type="submit" class="btn btn-primary" value="Submit Form">
```

This is add submit button to the form

3-Biding data with ngForm (cont.)

- Until now the form is a basic html form. Any time we use Form tag angular attaches **ngForm** directive to the form tag which gives it valuable information about the form.
- It gives what is the values of the different form control elements and are these values valid or not.

- **How to make reference to ngForm Directive?**

  By using Template Reference Variable

3-Biding data with ngForm (cont.)

- Using Template Reference Variable to reference ngForm.

```
   <form #userForm="ngForm">
   </form>
```

- The directive gives us access to the form control using the value property

```
{{userForm.value |json}}
```

3-Biding data with ngForm (cont.)

- If we look at the browser now we will see empty object **even if** we write values to the inputs elements.

- **Why it does not work??**
- If we ask angular why the value is an empty object it is going to tell you I’m not going to track every single form control in your html let me know which of the form controls have to be tracked then I will do that.

3-Biding data with ngForm (cont.)

- **Now ,How let angular know the controls to be tracked?**
- The answer is using **ngModel** directive.
- So to each of form controls place **ngModel** directive.
- **Such as the following**

```
<input type="tel" class="form-control“
  ngModel>
```

Note : We must add **name** property to each form controls elements.

3-Biding data with ngForm (cont.)

- In addition to **ngModel** directive angular provides the **ngModelGroup** directive we use it to group a collection of form control elements.
- Consider the address it contains street , city, state and postalCode.
- Let see the example.

3-Biding data with ngForm (cont.)

```
<div ngModelGroup="address">
    <div class="form-group">
          <label>Street</label>
          <input type="text" name="txtStrret" ngModel class="form-
            control" />
   </div>
   <div class="form-group">
          <label>City</label>
          <input type="text" name="txtCity" ngModel class="form-
            control" />
  </div>
   <div class="form-group">
          <label>Postal Code</label>
          <input type="text" name="txtPostalCode" ngModel
           class="form-control" />
  </div>
</div
```

3-Biding data with ngForm (cont.)

- In this section we will see how to bind data to a model.
- As user enter the form data we will capture the changes and update an instance of the model that can **be send to the server.**

- The first step is to generate **a model class**.
- Open command prompt
- Write the command

  ng g class className ex:User
- open User.ts and write class properties

3-Biding data with ngForm (cont.)

```
export class User {
    constructor(
            public name:string,
            public email:string,
            public phone:string,
            public street:string,
            public city:string,
            public PostalCode:string,
            public topic:string,
            public timeSelection:string,
            public chkOffers:boolean
)
{
}}
```

3-Biding data with ngForm (cont.)

This is TypeScript shorthand syntax of a constructor.

TypeScript compiler generates a public field for each public constructor parameter and automatically assigns the parameters value to that filed when you create a new user.

3-Biding data with ngForm (cont.)

- The second step is to create an instance of this model in **app.component.ts.**

```
userModel=new
User('Abanoub','abnabil@iti.gov.eg',111256,'Ass
uit','Assuit','7111','Angular','evening',true);
```

Let see how to bind user model data to enrollment form.

3-Biding data with ngForm (cont.)

- We will use interpolation to bind the user model and display it in the view
- In each form element ngModel will be replaced with the UserModel specific value

```
[(ngModel)]="userModel.name“
```

Now let us test the following

{{userModel|json}}

Questions

Any Questions?

THANK YOU ☺