CHIANG MAI UNIVERSITY

Bachelor of Science (Software Engineering)

College of Arts, Media and Technology

2nd Semester / Academic Year 2017

SE 331 Component Based Software Development

Separation and Data binding

Name ..... ID .....

Objective In this session, you will experience how to bind data from/to the view part, and tip and trick for the data binding

Suggestion you should read the instructions step by step. Please try to answer a question by question without skipping some questions which you think it is extremely difficult.

Hint The symbol + and – in front of the source code is to show that you have to remove the source code and add the source code only. There are not the part of the source code

0. Setting

   0.1. Open the folder `C:\lab`

   0.2. Open the command window and then go to the folder `C:\lab\`

   0.3. Type `git clone https://github.com/chartchai/SE331-lab03  lab03`

   0.4. Run `npm install` and `start` to check that you get the correct application

1. Event binding

   Event binding is to bind the event to the view part to the control part.

   1.1. Add the method to handle the event in the `students.component.ts` as given

```
            return sum/this.students.length;
        }
+
+       upQuantity(){
+           alert("You called upQuantity");
+       }
 }
```

   1.2. And then add the button to send the event, bind it with the `upQuantity` method.

```
            <p *ngIf="student.gpa > 2.5">Good Student who get grade
                {{student.gpa | number:'1.2-2'}}</p>
            <p *ngIf="student.gpa <= 2.5">Bad Student who get grade
                {{student.gpa  | number:'1.2-2'}}</p>
+           <div class="row">
+               <button type="button" class="btn btn-primary btn-xs col-
                md-1" (click)="upQuantity()">+</button>
+           </div>
            </div>
```

   Try to press the button what happen?

Lecture's Signature _____

1.3. Add the Quantity of the pen each student has,

In the `student.ts` add the new attributes

```
  featured:boolean;
-
+       penAmount:number;
 }
```

And then add the `penAmount` to each students in the `mocks.ts`

1.4. Add the part to show the amount of pen of the students in the `students.component.html`

```
                <div class="row">
-                   <button type="button" class="btn btn-primary btn-xs col-
                    md-1" (click)="upQuantity()">+</button>
+                   <div class="col-md-1">
+                    {{student.penAmount}}
+                   </div>
+                   <button type="button" class="btn btn-primary btn-xs col-
                    md-1" (click)="upQuantity(student)">+</button>
                </div>
```

1.5. The `upQuantity` method has been changed to receive the student. The input student's `penAmount` will be changed.

```
-       upQuantity(){
-               alert("You called upQuantity");
+       upQuantity(student){
+               student.penAmount++;
        }
```

Try to run the application, what happen when you click add button

1.6. Add the decrease button and the method binding to decrease the value.

In `students.component.html`

```
                <div class="row">
+                   <button type="button" class="btn btn-primary btn-xs col-
                    md-1" (click)="downQuantity(student)">-</button>
                    <div class="col-md-1">
```

In `students.component.ts`

```
+
+       downQuantity(student){
+               student.penAmount--;
+       }
```

Run the application, try to press the decrease button from the beginning.

1.7. The quantity should not be negative, modified the source code to prevent the negative quantity

Lecture's Signature _____

1.8. Create the reset button which will reset the pen amount into 0;

Lecture's Signature _____

1.9. The event can be injected as the input of the method

In the `students.component.html`

```
                    <button type="button" class="btn btn-primary btn-xs col-
                    md-1" (click)="downQuantity(student)">-</button>
-                   <div class="col-md-1">
+                   <div class="col-md-1" (mouseover)="getCoord($event)">
                                            {{student.penAmount}}
```

In the `students.component.ts`

```
+
+       getCoord(event){
+               console.log(event.clientX + ", " + event.clientY);
+       }
```

And then open the developer console by right click and select inspect element.

Select the console tab; you can see the console change when the mouse passes a number of pens

2. Two ways-binding

   Two ways-binding is used with the field with the input information to link the data from the view part to the control part directly.

   2.1. Add The text box by removing the text to show the amount of pen and replace it with the text box.

```
                <div class="row">
-                   <button type="button" class="btn btn-primary btn-xs col-
                    md-1" (click)="downQuantity(student)">-</button>
-                   <div class="col-md-1" (mouseover)="getCoord($event)">
-                                                       {{student.penAmount}}
-                   </div>
+                   <button type="button" class="btn btn-primary btn-xs col-
                    md-1" (click)="downQuantity(student)">-</button>
+                   <input type="text" class="col-md-1"
                    [value]="student.penAmount">
                    <button type="button" class="btn btn-primary btn-xs col-
                    md-1" (click)="upQuantity(student)">+</button>
                </div>
```

   2.2. The look and feel are not good so update the `students.comopnent.css` file.

Hint you may see the change after finish adding each line of the CSS

```
.featured{
        background: linear-gradient(to bottom right, red, yellow);
+}
+button{
+       box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2), 0 6px 20px 0
rgba(0,0,0,0.19);
+       margin-top: 2px;
+}
+input{
+       box-sizing: border-box;
+    border: 2px solid #ccc;
+    border-radius: 4px;
+       margin-left: 2px;
+       margin-right: 2px;
 }
```

   2.3. After finished update the css, try to type some number and click increase, or decrease. What happen and why?

.....

.....

   2.4. Now bind the text box with the `penAmount` attribute.

In `app.module.ts` import the `FormsMoudule` to the project.

```
import {TimeComponent} from './time/time.component';
```

```
+import {FormsModule} from '@angular/forms'
 @NgModule({
  declarations: [ AppComponent,
                                StudentsComponent,
                                TimeComponent],
- imports: [BrowserModule],
+ imports: [BrowserModule,FormsModule],
  bootstrap: [AppComponent]
 })
```

With the imports you can used the `FormsModule` throughout the project.

In `students.component.html`

```
                    <button type="button" class="btn btn-primary btn-xs col-
                    md-1" (click)="downQuantity(student)">-</button>
-                   <input type="text" class="col-md-1"
                    [value]="student.penAmount"/>
+                   <input type="text" class="col-md-1"
                    [(ngModel)]="student.penAmount"/>
                    <button type="button" class="btn btn-primary btn-xs col-
                    md-1" (click)="upQuantity(student)">+</button>
```

Try to type the value, and click the increase or decrease button, what it changes?

.....

.....

2.5. Where does the `ngModel` come from?

.....

.....

Lecture's Signature _____

3. Dependency injection: How to inject the dependency in the Angular

3.1. Create the data service component, to provide the data for the application. Create the `services` folder in the `app` folder, and then create a new file name `students-data.service.ts` with the given information.

```
+import {STUDENTS} from '../mocks';
+
+export class StudentDataService{
+       getStudentsData(){
+               return STUDENTS;
+       }
+}
```

3.2. Update the `students.component.ts` to get the Students information from the new service instead of the `mocks.ts`

```
import {Student} from './student';
-import {STUDENTS} from '../mocks';
+import {StudentDataService} from '../services/students-data.service';
 @Component({



        ngOnInit(){
-               this.students = STUDENTS;
+               let studentDataService = new StudentDataService();
+               this.students = studentDataService.getStudentsData();
        }
```

Run the application to see the result.

3.3. Setting the injector so that the object can be injected by the Angular.

Add the Injectable in the `students-data.service.ts`

```
+import {Injectable} from '@angular/core';
+@Injectable()
 export class StudentDataService{
        getStudentsData(){
                return STUDENTS;
```

Define the constructor for the `students.component.ts` to injection from the framework

```
        students: Student[];
+       constructor(private studentDataService:StudentDataService ){}
        ngOnInit(){
-               let studentDataService = new StudentDataService();
-               this.students = studentDataService.getStudentsData();
+
+               this.students = this.studentDataService.getStudentsData();
        }
```

Setup the injection in the `app.module.ts`

```
import {FormsModule} from '@angular/forms'
+import {StudentDataService} from './services/students-data.service';
 @NgModule({
  declarations: [ AppComponent,
                                StudentsComponent,
                                TimeComponent],
  imports: [BrowserModule,FormsModule],
- bootstrap: [AppComponent]
+ bootstrap: [AppComponent],
+ providers: [StudentDataService]
 })
```

Run the application, now you must inject the `StudentDataService` from the `app.module`

3.4. Change the injection data to use the other class.

In the `mocks.ts` add a new arrays of the students with 2 or 3 students and name it as `STUDENTS2`

In the service folder, create the `students-data2.service.ts` with the given information to return the

`STUDENTS2` array.

```
+import {STUDENTS2} from '../mocks';
+import {Injectable} from '@angular/core';
+@Injectable()
+export class StudentData2Service {
+       getStudentsData(){
+               return STUDENTS2;
+       }
+}
```

Update the `app.module.ts` to inject the `StudentData2Service` as given

```
import {StudentDataService} from './services/students-data.service';
+import {StudentData2Service} from './services/students-data2-file.service';
 @NgModule({
```

```
  bootstrap: [AppComponent],
- providers: [StudentDataService]
+ providers: [{provide:StudentDataService,useClass:StudentData2Service}]
```

```
})
```

Show the result

Lecture's Signature _____

4. Using the `http` component, the `http` component is proposed so that the application can communicate with the server.

4.1. Import the `http` module

```
import {StudentDataFileService} from './services/students-data-file.service';
+import {HttpModule} from '@angular/http';
 @NgModule({
  declarations: [ AppComponent,
                                StudentsComponent,
                                TimeComponent],
- imports: [BrowserModule,FormsModule],
+ imports: [BrowserModule,FormsModule, HttpModule],
  bootstrap: [AppComponent],
```

4.2. Create the data file to be loaded, create the `data` folder in the `app` folder. Create the `student.json`

```
+{
+       "data":[
+               {
+                       "id": 1,
+                       "studentId":"562110509",
+                       "name":"Pu",
+                       "surname":"Priya",
+                       "gpa":0,
+                       "image":"images/pu.jpg",
+                       "featured":false,
+                       "penAmount":0
+               },
+               {
+                       "id": 2,
+                       "studentId":"562110507",
+                       "name":"Prayuth",
+                       "surname":"Tu",
+                       "gpa":4.00,
+                       "image":"images/tu.jpg",
+                       "featured":true,
+                       "penAmount":0
+               }
+       ]
+}
```

Hint this is the same information just swap the first student, and the second student.

4.3. Add the `students-data-file.service.ts` to the `service` folder with the given information.

```
import {Injectable} from '@angular/core';
import {StudentDataService} from './students-data.service';
import {Http} from '@angular/http';
import {Student} from '../students/student';
import 'rxjs/add/operator/map';
@Injectable()
export class StudentDataFileService {
        constructor(private http: Http){}
        getStudentsData(){
                return this.http.get('app/data/student.json')
                        .map(res => res.json().data);
        }
}
```

4.4. Now you get the `Observable` object from the `getStudentData()`, parse it to the `Student` array object in the `students.component.ts`

```
+import {StudentDataFileService} from '../services/students-data-file.service';
```

And update the class

```
-       constructor(private studentDataService:StudentDataService ){}
+       constructor(private studentDataService:StudentDataFileService ){}


        ngOnInit(){

-               this.students = this.studentDataService.getStudentsDatastudentDataServicegetStudentsData();
+
+               this.studentDataService.getStudentsData()
+                       .subscribe(data => this.students = data,
+                                  error => alert("Error is : " + error),
+                                  ()=>console.log("finished")
+                       );
+
        }
```

4.5. Change the provider in the app.module.

```
  bootstrap: [AppComponent],
- providers: [{provide:StudentDataService,useClass:StudentDataFileService}]
+ providers: [StudentDataFileService]
```

4.6. Now, run the application.

4.7. You may notice that there is an error as the students may not be an array yet while the `averageGpa()` is called. To prevent the error, check whether the students is an array or not before reading it.

```
        averageGpa(){
                let sum = 0;
-               for(let student of this.students){
-                       sum += student.gpa;
                }
-               return sum/this.students.length;return sum/this.students.length;

+               if (Array.isArray(this.students)){
+                       for(let student of this.students){
+                               sum += student.gpa;
+                       }
+                       return sum/this.students.length;
+               }else{
+                       return 0.0;
                }
        }
```

Lecture's Signature _____

5. Create a new service class; this class must download the information from https://s3-ap-southeast-1.amazonaws.com/se331/people.json and display to the website. The injection must be used; in other words, the `students.component.ts` must not be updated. In addition, there are some UI, which are not look good. Update the CSS file to make it show all the information properly.

Lecture's Signature _____