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-lab04  lab04`

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

0.5. Try to use the ide to develop the application, download the latest `Intellij idea`, and then apply for Jetbrain student pack at `https://www.jetbrains.com/student/` and then open the application

0.6. Install the AngularJS plugins by File->Setting ->Plugins. Click install Jetbrains Plugins and search for AngularJS plugin to install.

0.7. Select `File->new-> from Existing` Code and then select the `lab04` folder to start the application

0.8. Now in app folder, there are three components in the student folder, which are students list, student add, and student view

Hint The default of the path strategy in Angular is the path strategy; however, it may not be support some feature via the lite-server. As a consequence, to complete some operation in our development environment, the hash path strategy may be applied by add the given provider in the provides part.

```
{ provide: LocationStrategy, useClass: HashLocationStrategy }
```

1. Routing

1.1. Routing is the technique to change the component dynamically via the menu or the simple link.

Event binding is to bind the event to the view part to the control part. In this part, the menu component will be created by creating the `menu` folder in the `app` folder

And then add the given file in the `menu` folder

`menu.component.html`

```
+<ul class="nav nav-pills nav-stacked" >
+  <li role="presentation" class="active"><a href="#">List</a></li>
```

```
+  <li role="presentation"><a href="#">Add</a></li>
+  <li role="presentation"><a href="#">View</a></li>
+</ul>
```

menu.component.ts

```
+import {Component} from '@angular/core';
+@Component({
+ selector: 'menu',
+ templateUrl: 'app/menu/menu.component.html',
+ styleUrls:['app/menu/menu.component.css']
+})
+export class MenuComponent {
+
+}
```

menu.component.css

// leave blank

1.2. Add the menu component to the `app.module.ts`

```
          StudentsViewComponent,
-                                TimeComponent],
+                                TimeComponent,
+         MenuComponent],
  imports: [BrowserModule,FormsModule, HttpModule],
```

Hint with the ide, when you add the `MenuComponent`, it may auto import the reference for you automatically, or if it not, it will show the red underline under the `MenuComponent` you can right click to ask it to auto import the component. This will help you to write you code efficiently.

1.3. Update the `app.component.html` to show the menu

```
<div class="container-fluid">
-<div class="row page-header">
-<h1 class="col-md-offset-2">Hello {{name}}</h1>
-</div>
-<div class="row">
-       <students></students><br/>
-</div>
-<div class="row">
-</div>
   <div class="row page-header">
     <h1 class="col-sm-offset-1">Hello {{name}}</h1>
   </div>
+  <div class="row">
+    <div class="col-sm-3 col-md-2">
+      <menu></menu>
+    </div>
+    <div class="col-sm-9 col-md-8">
+      <students></students>
+    </div>
+  </div>
+  <div class="row">
+  </div>
 </div>
```

Now look at your application, the menu is shown.

1.4. With the menu, the students list look not good, consider to remove the `col-md-offset` at the given location in the `students.component.html`

```
-              <div class="panel panel-primary col-md-offset-2 col-md-6 col-
sm-offset-1 col-sm-8" >
+              <div class="panel panel-primary col-md-6 col-sm-8" >
```

```
-       <div class="alert alert-success col-md-offset-2 col-md-6"> The average
gpa is {{averageGpa()|number:'1.2-2'}} </div>
+       <div class="alert alert-success col-md-6 col-sm-8"> The average gpa is
{{averageGpa()|number:'1.2-2'}} </div>
```

How does it look?

What is the `*-offset` for?

.....

.....

1.5. To add the routing function, the base tag must be defined in the `index.html` in the head element

```
+    <base href="/">
```

1.6. The Router module, and its setting must be added in `app.module.ts`

```
+const appRoutes: Routes = [
+  {path: 'view',component: StudentsViewComponent},
+  {path: 'add', component: StudentsAddComponent},
+  {path: 'list', component: StudentsComponent}
+];

 @NgModule({
- imports: [BrowserModule,FormsModule, HttpModule],
+ imports: [BrowserModule,FormsModule, HttpModule,
+          RouterModule.forRoot(appRoutes)],
  bootstrap: [AppComponent],
```

again, you may use the auto import to find where is the component location.

1.7. Where is the `StudentsViewComponent`, `StudentsAddComponent`, and `StudentsComponent` located?

.....

.....

1.8. Prepare the place which the component will be replaced, update the app.component.html

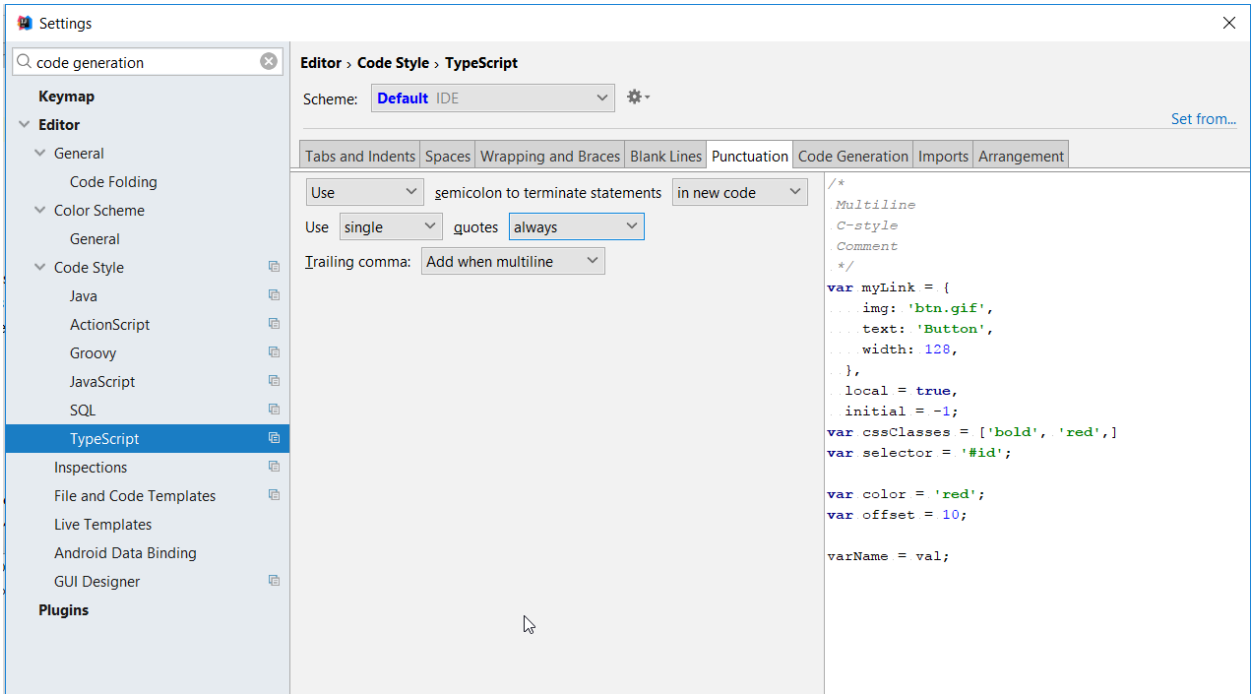
```
    <div class="col-sm-9 col-md-8">
-     <students></students>
+     <router-outlet></router-outlet>
    </div>
  </div>
```

Now you can browse to all the page by browsing at `localhost:3000(` or `localhost:3000/#` if you use the hash path location strategy) `/view`, `/add`, and `/list`. Update the HTML to make your application look better.

Lecture's Signature _____

Hint If you find the error from TSLint which refer to the format of your source code. Try Ctr-Alt-L to fix the layout. This will make your code look better.

Hint You may find the autogeneration error on TSLint as they suggest to use single quote (') for declaring the literacy in TS. You may update your IntelliJ preference by open the project setting and set the IntelliJ Setting as shown here



2. Adding the link

2.1. The link to each component can be added, edit the `menu.component.html` to add the link as given

```
<ul class="nav nav-pills nav-stacked" >
-   <li role="presentation" class="active"><a href="#">List</a></li>
-   <li role="presentation"><a href="#">Add</a></li>
-   <li role="presentation"><a href="#">View</a></li>
+   <li role="presentation" class="active"><a routerLink="/list">List</a></li>
+   <li role="presentation"><a routerLink="/add">Add</a></li>
+   <li role="presentation"><a routerLink="/view">View</a></li>
 </ul>
```

Try to click on the link

2.2. To make the active class applied when clicking the link, update the `menu.component.html`

```
<ul class="nav nav-pills nav-stacked" >
-   <li role="presentation" class="active"><a routerLink="/list">List</a></li>
-   <li role="presentation"><a routerLink="/add">Add</a></li>
-   <li role="presentation"><a routerLink="/view">View</a></li>
+   <li role="presentation" routerLinkActive="active"><a
routerLink="/list">List</a></li>
+   <li role="presentation" routerLinkActive="active"><a
routerLink="/add">Add</a></li>
+   <li role="presentation" routerLinkActive="active"><a
routerLink="/view">View</a></li>
 </ul>
```

Try to click on the link

3. Sometime, the user may type the url to the resource which we have not provided, not try to browse to `/abc` location, what is shown on the screen?

.....

.....

3.1. Create the `FileNotFoundComponent` component for handle the URL request which is not provided in our application by creating the `filenotfound` folder in the app folder

3.2. Create the `file-not-found-component.ts` using the given code

```
+import {Component} from '@angular/core';
+@Component({
+ selector: 'file-not-found',
+ templateUrl: 'app/filenotfound/file-not-found.component.html',
+ styleUrls:['app/filenotfound/file-not-found.component.css']
+})
+export class FileNotFoundComponent {
+
+}
```

3.3. Create the file-not-found-component.html using the given code

```
+<H2>The resource you have asked is not in the server</H2>
+<img src="./app/filenotfound/filenotfound.jpg">
```

You have to find the file by yourselves and leave the `CSS` file blank for the further purpose.

3.4. Declare the `FileNotFoundComponent` in the `app.module.ts`

3.5. Add the path reception to the `appRoutes`

```
    {path: 'add', component: StudentsAddComponent},
-   {path: 'list', component: StudentsComponent}
+   {path: 'list', component: StudentsComponent},
+   {path: '**', component: FileNotFoundComponent}
  ];
```

Now try to browse to somewhere else

Lecture's Signature _____

4. Set welcome page

4.1. Now try to type only `localhost:3000` what is shown?

.....

.....

4.2. To change the default location add this to the `appRoutes`

```
    {path: 'list', component: StudentsComponent},
+   { path: '',
+     redirectTo: '/list',
+     pathMatch: 'full'
+   },
    {path: '**', component: FileNotFoundComponent}
```

Then try again

4.3. Now change the welcome page to the add page

Lecture's Signature _____

5. Attached Data

The data can be provided in the `appRoutes` so that it can be display in the page.

5.1. Add the data to the `/view` to be shown in the `StudentViewComponent`

```
-   {path: 'view',component: StudentsViewComponent},
+   {
+     path: 'view', component: StudentsViewComponent
+     ,
+     data: {
+       student: {
+         "id": 2,
+         "studentId": "SE-001",
+         "name": "Prayuth",
```

```
+         "surname": "The minister",
+         "gpa": 3.59,
+         "image": "images/tu.jpg",
+         "featured": false,
+         "penAmount": 15,
+         "description": "The great man ever!!!!"
+       }
+     }
+   },
```

Hint the data is provided in the `data/people.json` you can copy it and paste.

You may need to add the description attribute in Student class. Do not forget to add something in the mock.ts to make the application can be compiled.

5.2. Update the `students.view.component.html` to show the data from the input student

```
-         //temp for id
+         {{student.id}}

-         // temp for name
+         {{student.name}} {{student.surname}}

-         0
+         {{student.penAmount}}

-         // wait for file location
+       <div class="col-md-4 col-sm-6">
+         <img class="img-responsive" [src]="student.image" [alt] =
"student.image">
        </div>

-         <p>// wait for the description</p>
+         <p>{{student.description}}</p>
```

5.3. The `students.view.component.ts` needs the `ActivatedRoute` to read the data from the `appRoutes` as given

```
export class StudentsViewComponent {
-       students: Student[];
+   constructor(private route: ActivatedRoute) {}
+    student:Student;
+    ngOnInit() {
+       this.route
+       .data
+       .subscribe(inputData=>{this.student =(inputData as Any).student;});
+   }
```

Hint to make the source code more beautiful; you can press `Ctrl-Alt-L` to auto layout your source code.

Now see the `/view`

6. Extract the routing information

6.1. To make the `app.module.ts` clearer, the routing information can be extracted to other module.

Create the `app-routing.module.ts` and copy all the routing information from the old `app.module` as given.

```
+const appRoutes: Routes = [
+  {
+    path: 'view', component: StudentsViewComponent
+    ,
+    data: {
+      student: {
+        "id": 2,
+        "studentId": "SE-001",
+        "name": "Prayuth",
+        "surname": "The minister",
+        "gpa": 3.59,
+        "image": "images/tu.jpg",
+        "featured": false,
+        "penAmount": 15,
+        "description": "The great man ever!!!!"
+      }
+    }
+  },
+  {path: 'add', component: StudentsAddComponent},
+  {path: 'list', component: StudentsComponent},
+  {
+    path: '',
+    redirectTo: '/list',
+    pathMatch: 'full'
+  },
+  {path: '**', component: FileNotFoundComponent}
+];
+
+@NgModule({
+  imports: [
+    RouterModule.forRoot(appRoutes)
+  ],
+  exports: [
+    RouterModule
+  ]
+})
+export class AppRoutingModule {
+}
```

6.2. In `app.module.ts`, remove the `appRoutes` information, and imports the `AppRoutingModule` as given

```
  imports: [BrowserModule, FormsModule, HttpModule,
-    RouterModule.forRoot(appRoutes)],
+   AppRoutingModule],
  bootstrap: [AppComponent],
```

try to routing the application

Lecture's Signature _____

6.3. To set the students as a set of component, we can extract only the routing information for the student components. Create the `student-routing.module.ts` in the `students` folder, and then copy only the students' components routing information to a new file as given

```
+const studentRoutes: Routes = [
+  {
+    path: 'view', component: StudentsViewComponent
+    ,
+    data: {
+      student: {
+        "id": 2,
+        "studentId": "SE-001",
+        "name": "Prayuth",
+        "surname": "The minister",
+        "gpa": 3.59,
```

```
+        "image": "images/tu.jpg",
+        "featured": false,
+        "penAmount": 15,
+        "description": "The great man ever!!!!"
+      }
+    }
+  },
+  {path: 'add', component: StudentsAddComponent},
+  {path: 'list', component: StudentsComponent},
+  {
+    path: '',
+    redirectTo: '/list',
+    pathMatch: 'full'
+  }
+];
+
+@NgModule({
+  imports: [
+    RouterModule.forRoot(studentRoutes)
+  ],
+  exports: [
+    RouterModule
+  ]
+})
+export class StudentRoutingModule {
+}
```

6.4. Update the AppRoutingModule; only FileNotFoundComponent is still in the appRoutes.

6.5. Import the StudentRoutingModule to the app.module.ts as given

```
-    AppRoutingModule],
+    AppRoutingModule,StudentRoutingModule],
```

show the routing

6.6. The File not found page is shown, fix it to the normal application

Lecture's Signature _____

7. Provide the mocking database

7.1. To provide the simple database operation, the service which can keep the data and update data is required so create the `students-data-db.service.ts` in the service folder

```
+@Injectable()
+export class StudentsDataDBService {
+
+  students:Student[] = [
+    {
+      "id": 1,
+      "studentId":"SE-001",
+      "name":"Prayuth",
+      "surname":"The minister",
+      "gpa":3.59,
+      "image":"images/tu.jpg",
+      "featured":false,
+      "penAmount":15,
+      "description":"The great man ever!!!!"
+    },
+    {
+      "id": 2,
+      "studentId":"SE-002",
+      "name":"Jurgen",
+      "surname":"Kloop",
+      "gpa":2.15,
+      "image":"images/Kloop.gif",
```

```
+      "featured":true,
+      "penAmount":2,
+      "description":"The man for the Kop"
+    },
+    {
+      "id": 3,
+      "studentId":"SE-003",
+      "name":"Mitsuha",
+      "surname":"Miyamizu",
+      "gpa":2.15,
+      "image":"images/mitsuha.gif",
+      "featured":true,
+      "penAmount":0,
+      "description" : "The most beloved one"
+    }
+  ]
+}
```

Hint the data is already in the `people.json`, you can copy from there directly.

7.2. As the old provider provide the `Observable` object, the new service must return the `Observable` object, too. In the `students-data-db.service.ts` import this two components

```
import 'rxjs/add/operator/map';
+import {Observable} from "rxjs/Observable";
+import {Subscriber} from "rxjs/Subscriber";
 @Injectable()
```

And then add the `getStudentsData()` function

```
   getStudentsData(){
+    return new
Observable<Student[]>((subscriber:Subscriber<Student[]>)=>subscriber.next(thi
s.students));
   }
```

The parameter in `subscriber.next()` will be the object that the subscribe method will get

Now the application should run properly.

7.3. Inject `StudentDataDBService` instead of `StudentDataService` by updating `app.module.ts`

7.4. The path `id`, the `id` value can be passed to the Component, for example, `detail/15`. The number `15` can be passed to the component by add the details mapping in the `student-routing.module.ts`

```
-    path: 'view', component: StudentsViewComponent
+  {path: 'detail/:id',component:StudentsViewComponent}
```

7.5. This will show the information of the id which is given in the URL, the service should provide the method to find the student by id. Add the add function in the `StudentsDataDBService` component

```
+  getStudent(id:number){
+    let student = this.students.find(student=> student.id === +id);
+    return new
Observable<Student>((subscriber:Subscriber<Student>)=>subscriber.next(student
));
+  }
```

However, as the `StudentsDataDBService` is the provider for the `StudentsDataService`, so the `StudentsDataService` must contains the same method, but it can be method with a null as given.

```
+  getStudent(id:number){
+   return null;
+  }
```

7.6. Update the `students.view.component.ts`, import the required function as given

```
+import 'rxjs/add/operator/switchMap';
```

Inject the `StudentDataService` to the `StudentsViewComponent`

```
+  constructor(private route: ActivatedRoute, private
       studentDataService:StudentsDataService) {}
```

Change the load on ngOnInit

```
    ngOnInit() {
-       this.route
-       .data.subscribe(inputData=>{this.student = (inputData as
any).student;});
+       this.route.params
+         .switchMap((params:Params) =>
this.studentDataService.getStudent(+params['id']))
+         .subscribe((student:Student) => this.student = student);
```

If you have not change the `locationStrategy` to the `Hash Location Strategy`, you may find the error, and the program cannot be loaded. If that happens, try to change the `LocationStrategy`

Now browse the by `/detail/2` to see what happened

Lecture's Signature _____

7.7. Now try to browse `detail/8`, there will be errors as we do not have the students with id 8, to prevent this error, we move the browsing to the list. The users should click at each people in the list to see their details. In `StudentsComponent`, inject Router as the router as given

```
-  constructor(private studentDataService: StudentsDataService) {
+  constructor(private studentDataService: StudentsDataService, private
router: Router ) {
```

7.8. Create the `showDetail` function to redirect to details page, with the id to be shown

```
+  showDetail(student: Student){
+    this.router.navigate(['/detail',student.id]);
+  }
```

7.9. Update `students.component.html` to receive the clicked event and redirect to the detail page

```
      <div *ngFor="let student of students" class="row">
-             <div class="panel panel-primary col-md-8 col-sm-8" >
+             <div class="panel panel-primary col-md-8 col-sm-8"
(click)="showDetail(student)" >
                          <div class="panel-heading">
```

7.10. Create the drop down menu which will show the list of the students by name, and when click on the student's name, it will lead to the details page.

Hint you can look at `http://getbootstrap.com/components/#pills-with-dropdowns` to do the drop-down menu.

Lecture's Signature _____

8. Add object

8.1. To add object, the `StudentsDataDBService` must provide the `addStudent` function as given

```
+  addStudent(student:Student){
+    student.id = this.students.length+1;
```

```
+     this.students.push(student);
+   }
```

The empty `addStudent` with the same parameter is still required on the `StudentsDataService` class as the provider

8.2. To add data, the 2 ways-binding is required in the `students.add.component.html` as given

```
-         <input type="text" class="form-control" id="studentId"
placeholder="Student id">
+         <input type="text" class="form-control" id="studentId"
placeholder="Student id" [(ngModel)]="student.studentId"
+                name="id">


-         <input type="text" class="form-control" id="studentName"
placeholder="Name">
+         <input type="text" class="form-control" id="studentName"
placeholder="Name" [(ngModel)]="student.name"
+                name="name">


-         <input type="text" class="col-md-1 form-control input-amount"
value="0">
+         <input type="text" class="col-md-1 form-control input-amount"
value="0" [(ngModel)]="student.penAmount"
+                name="penAmount">


-         <textarea class="form-control" id="studentDescription"
placeholder="Description" row="3"></textarea>
+         <textarea class="form-control" id="studentDescription"
placeholder="Description" row="3"
+                   [(ngModel)]="student.description"
name="description"></textarea>
```

Note: the name attribute is required for the `ngModel`

8.3. Inject the router service, and `StudentDataService` and provide the students object to the link with the html

```
export class StudentsAddComponent {
-       students: Student[];
+       student: any= {};
+   constructor (private studentDataService:StudentsDataService, private
router: Router){};
+
```

8.4. The `addStudent` method must be provided in the component

```
+       addStudent(student:Student){
+               this.studentDataService.addStudent(student);
+               alert("Add complete");
+               this.router.navigate(['/list']);
+   }
```

8.5. Add the add button event to call the `addStudent` function

```
-         <button type="submit" class="btn btn-default">Add</button>
+         <button class="btn btn-default"
(click)="addStudent(student)">Add</button>
```

8.6. The input type"File" could not bind automatically with the Angular2 object, so add the function to handle the value as given

```
+   onFileChange(event,student:any){
+     var filename= event.target.files[0].name;
+     console.log(filename);
+     student.image=filename;
```

8.7. And then add the event handling for the file picker change value as given

```
-         <input type="file" class="btn btn-default" id="exampleInputFile">
+         <input type="file" class="btn btn-default" id="exampleInputFile"
(change)="onFileChange($event,student)" >
```

Now try to add a new value

Lecture's Signature _____