ROUTING

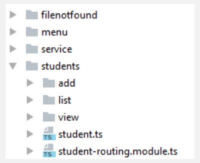
SE331 Component Based Software Development

REUSE COMPONENTS

- Angular create a components
  - As packed in the directory as each component

APPLICATION

- Reuse by calling the component to



```
▶ filenotfound
▶ menu
▶ service
▼ students
    ▶ add
    ▶ list
    ▶ view
    ▶ student.ts
    ▶ student-routing.module.ts
```

ROUTING

- Framework to call the implemented component to place in the Single page application

SINGLE PAGE APPLICATION

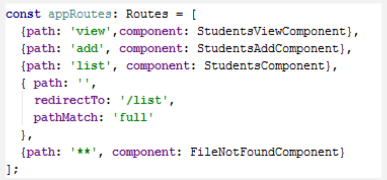
- Question 1 What is a Single Page Application

CONTROLLING THE COMPONENT

- CBSD must be able to config the framework
- The configuration file must not update the source code of the components
- Question 2 What are the components we used in the lab?

SET THE CONFIGURATION

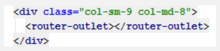
- The app.module as the configuration
- Create the path
- RegEx matching for the path
- Matching by order
- When Match
  - Call the component



```
const appRoutes: Routes = [
  {path: 'view',component: StudentsViewComponent},
  {path: 'add', component: StudentsAddComponent},
  {path: 'list', component: StudentsComponent},
  { path: '',
    redirectTo: '/list',
    pathMatch: 'full'
  },
  {path: '**', component: FileNotFoundComponent}
];
```

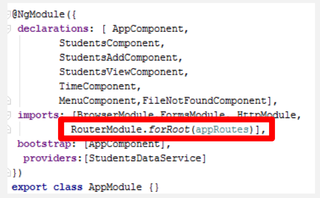
SETUP THE TARGET

- The target in the HTML file (app.component.html)
- Where the component will be placed



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

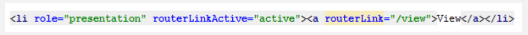
- Set up the route to the application



```
@NgModule({
 declarations: [ AppComponent,
         StudentsComponent,
         StudentsAddComponent,
         StudentsViewComponent,
         TimeComponent,
         MenuComponent,FileNotFoundComponent],
 imports: [BrowserModule,FormsModule, HttpModule,
          RouterModule.forRoot(appRoutes)],
 bootstrap: [AppComponent],
  providers:[StudentsDataService]
})
export class AppModule {}
```

ADD THE HTML LINK

- The angular tag provide the better link location tag
- Directive the tag that can be used in the Html



```html
<li role="presentation" routerLinkActive="active"><a routerLink="/view">View</a></li>
```

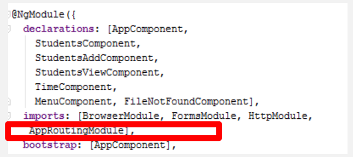
- Q4 where is the reference of the routerLinkActive

MODULARIZE THE PROJECT

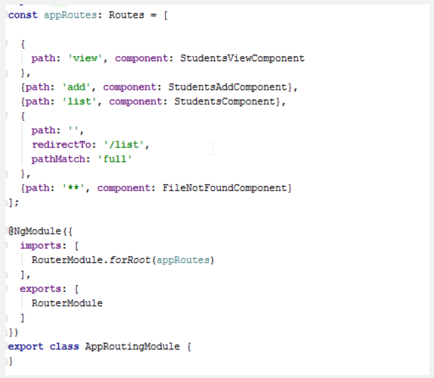
- Separate the concerns
- App.module => define the module to be used in the project
- The route information should not be there
- Question 3. where the route information should be? And why?

SEPARATE THE CONCERNS

- Move the information to the app.route
- And import it to the app.module



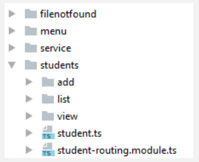
```typescript
@NgModule({
  declarations: [AppComponent,
    StudentsComponent,
    StudentsAddComponent,
    StudentsViewComponent,
    TimeComponent,
    MenuComponent, FileNotFoundComponent],
  imports: [BrowserModule, FormsModule, HttpModule,
   AppRoutingModule],
  bootstrap: [AppComponent],
```



```typescript
const appRoutes: Routes = [

  {
    path: 'view', component: StudentsViewComponent
  },
  {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 {
}
```

MAKING THE STUDENT COMPONENTS MORE MODULAR

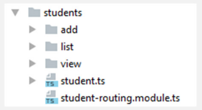
- Make the student components o be used easier in other project
- To make it simple
  - Only students package can be copy with the link information
- Move the link information to the package



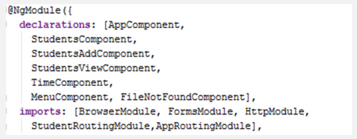
```
▶ filenotfound
▶ menu
▶ service
▼ students
    ▶ add
    ▶ list
    ▶ view
    ▶ student.ts
    ▶ student-routing.module.ts
```

MORE MODULARIZE COMPONENT

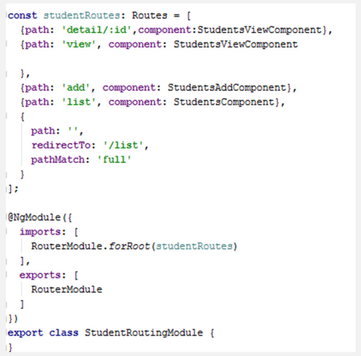
- Move the routing information to the package



- Import the configuration

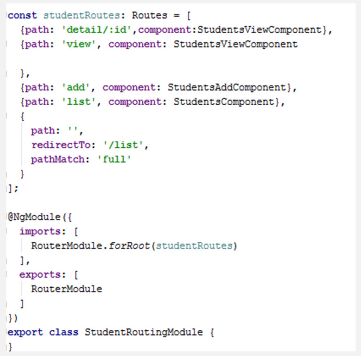


```
@NgModule({
  declarations: [AppComponent,
    StudentsComponent,
    StudentsAddComponent,
    StudentsViewComponent,
    TimeComponent,
    MenuComponent, FileNotFoundComponent],
  imports: [BrowserModule, FormsModule, HttpModule,
    StudentRoutingModule,AppRoutingModule],
```

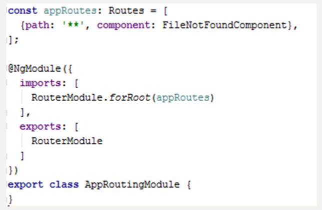


```
const studentRoutes: Routes = [
  {path: 'detail/:id',component:StudentsViewComponent},
  {path: 'view', component: StudentsViewComponent


  },
  {path: 'add', component: StudentsAddComponent},
  {path: 'list', component: StudentsComponent},
  {
    path: '',
    redirectTo: '/list',
    pathMatch: 'full'
  }
];

@NgModule({
  imports: [
    RouterModule.forRoot(studentRoutes)
  ],
  exports: [
    RouterModule
  ]
})
export class StudentRoutingModule {
}
```

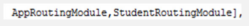
FIRST COME, FIRST SERVES



```
const studentRoutes: Routes = [
  {path: 'detail/:id',component:StudentsViewComponent},
  {path: 'view', component: StudentsViewComponent


  },
  {path: 'add', component: StudentsAddComponent},
  {path: 'list', component: StudentsComponent},
  {
    path: '',
    redirectTo: '/list',
    pathMatch: 'full'
  }
];

@NgModule({
  imports: [
    RouterModule.forRoot(studentRoutes)
  ],
  exports: [
    RouterModule
  ]
})
export class StudentRoutingModule {
}
```



```
const appRoutes: Routes = [
  {path: '**', component: FileNotFoundComponent},
];

@NgModule({
  imports: [
    RouterModule.forRoot(appRoutes)
  ],
  exports: [
    RouterModule
  ]
})
export class AppRoutingModule {
}
```

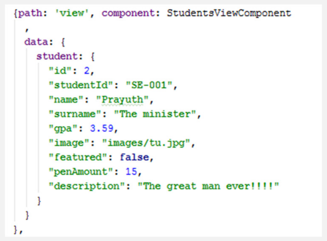


```
AppRoutingModule,StudentRoutingModule],
```

PRESET DATA IN THE ROUTING

- Components must be configurable
- Initial data can be inject via the routing component
- Send the data to the application

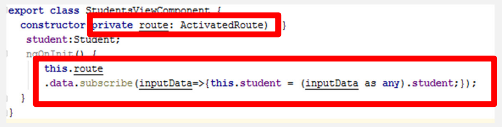
DATA INPUT AS THE OBSERVABLE OBJECT



```
{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!!!!"
    }
  }
},
```

RETRIEVE THE DATA

- As observable object



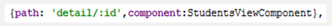
```
export class StudentsViewComponent {
  constructor(private route: ActivatedRoute) {}
   student:Student;
   ngOnInit() {
     this.route
     .data.subscribe(inputData=>{this.student = (inputData as any).student;});
  }
}
```

QUESTION

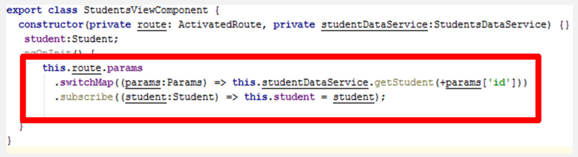
- How can we find the other parameters for the routing?

RETRIEVE PATH PARAMETER

- Knowing another component
- Reading the Path



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

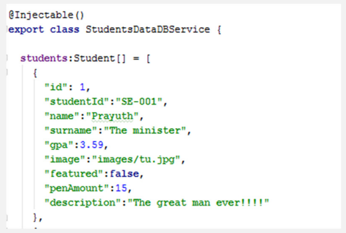


```
export class StudentsViewComponent {
  constructor(private route: ActivatedRoute, private studentDataService:StudentsDataService) {}
   student:Student;
   ngOnInit() {
     this.route.params
       .switchMap((params:Params) => this.studentDataService.getStudent(+params['id']))
       .subscribe((student:Student) => this.student = student);
  }
}
```

MOCKING THE OBSERVABLE OBJECT

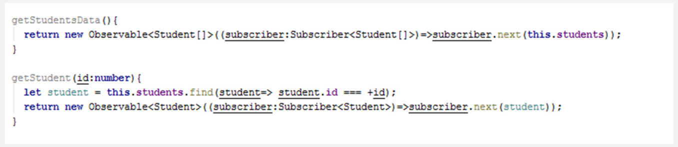
- The data service to be mocking
- Mocking for the real DB service
  - Which connects via Http object
  - Can be inject easily

USING ARRAY AS DATABASED



```
@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!!!!"
    },
```

RETURNING THE OBSERVABLE OBJECT



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

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

LOCATION STRATEGY

- HTML 5 pushState
  - Change a location without triggering a server page request
  - "natural" url
  - Newer
- Hash URL (#)
  - The request will be handle after a "#"
  - Location and path must be after the hash

localhost:3000/students

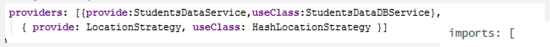
localhost:3000/#/students

ANGULAR2 PROVIDER

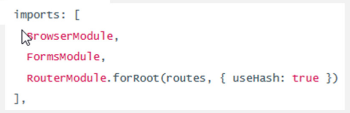
- PathLocationStrategy
  - Default style
  - Set for the "HTML 5 pushState" style
- HashLocationStrategy
  - The "hash URL" style
  - To use, we need to inject it

USING HASH URL

- Change the providers



```
providers: [{provide:StudentsDataService,useClass:StudentsDataDBService},
  { provide: LocationStrategy, useClass: HashLocationStrategy }]
```



```
imports: [
  BrowserModule,
  FormsModule,
  RouterModule.forRoot(routes, { useHash: true })
],
```

- Why
  - The Lite Server is not support the HTML push state when using the path value
  - Check your server later

SHOW THE LINK OF ALL COMPONENTS

- Including
  - ActiveRoutes
  - Routes
  - And any other component in the TS file

Q/A