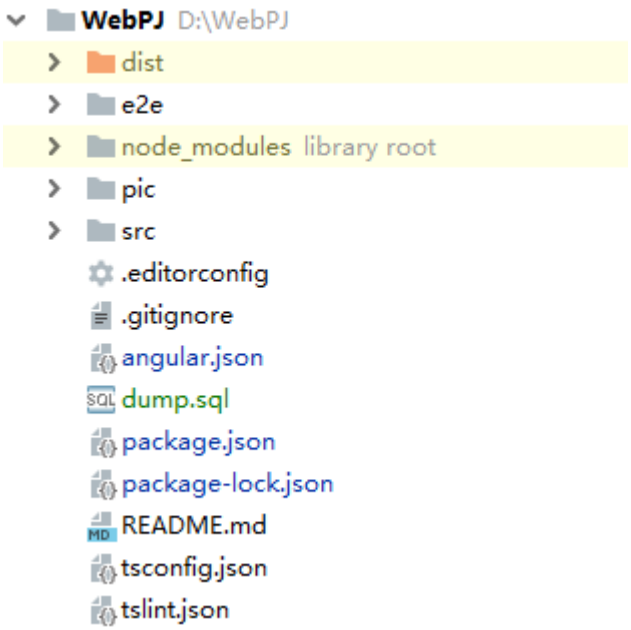
PJ 项目设计文档

易旭日小组

一、 教师端项目结构:

项目整体结构:



dist 文件夹中是 angular 打包后生成的文件夹, 用于部署到服务器的 nginx 上。

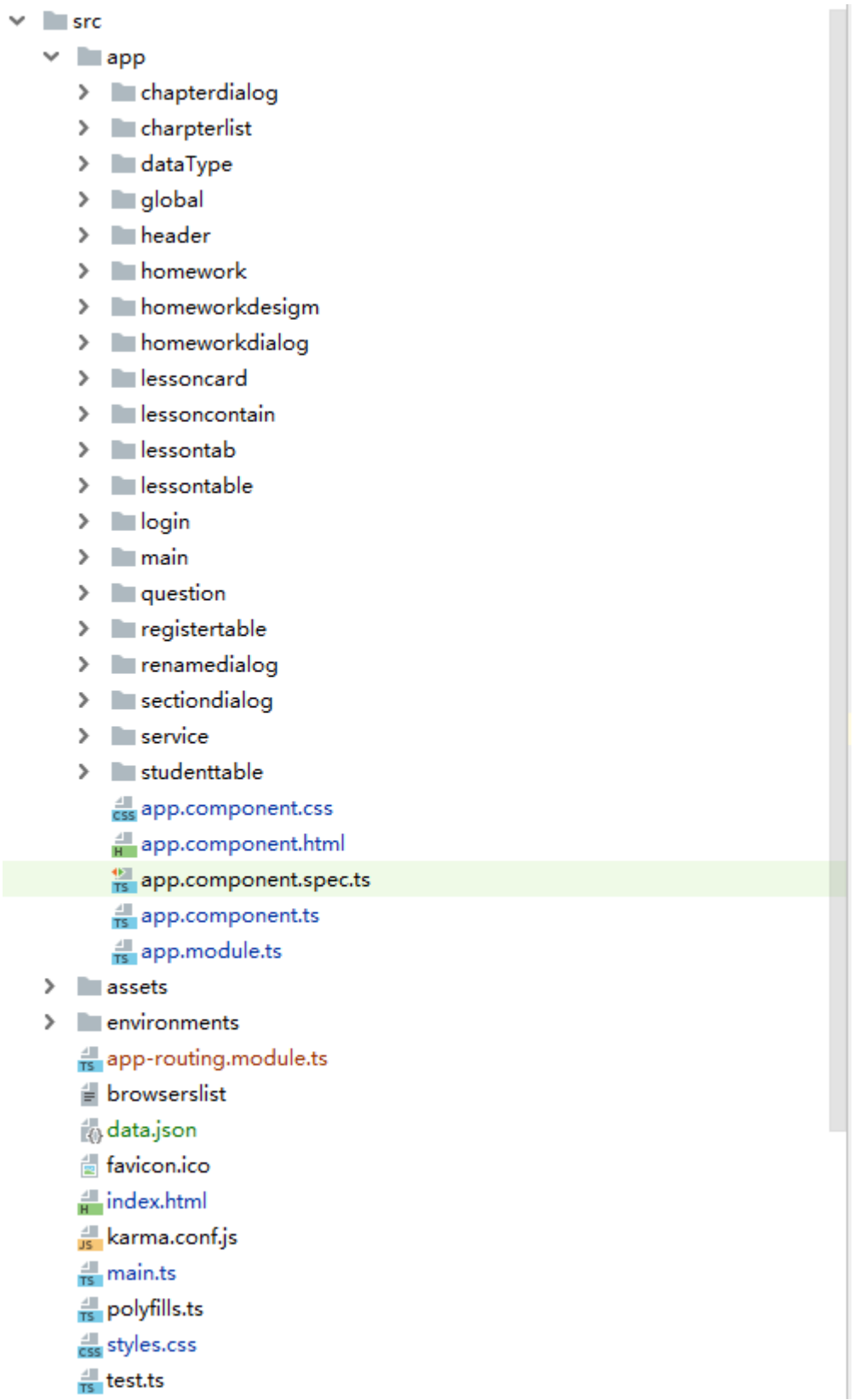
E2e 以及 node_modules 文件夹是 angular 自带的文件夹，和 PJ 核心功能关系并不是很大。

Pic 文件夹用于存放本次 PJ 用到的图片。

Src 文件夹在之后详细说明。

Src 文件夹之后的文件是 angular 自带的一些文件，也和 PJ 核心功能关系并不是很大，在此不做说明。

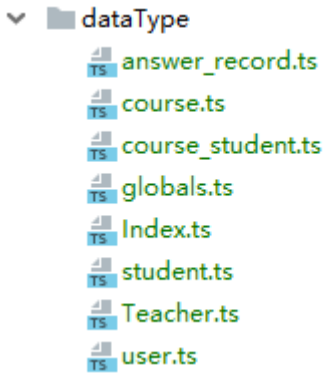
Src 目录结构:



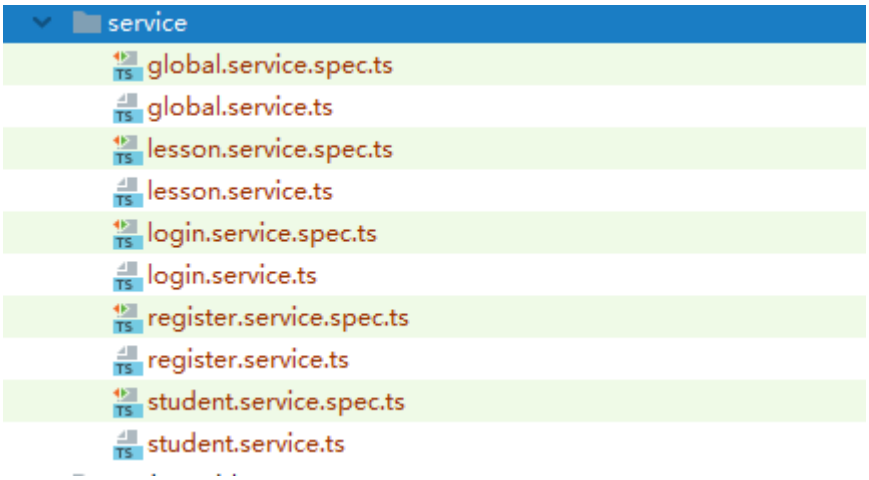
这是 src 文件夹下的内容，主要是项目的模块信息。

例如 lessoncard、registertable 文件夹等都是用于构建 angular 项目页面的模块内容，在此不做详细说明。

dataType 文件夹用于存放项目中所用到的数据类型，例如学生、老师等数据类型。



Service 文件夹是 angular 项目的 Service 部分，主要用于与后台进行交互，例如获取老师的开课信息等。



App.component.html 是 angular 项目的 html 文件，该项目是一个单页面应用，通过动态改变该 html 内容实现页面跳转。

App.modules.ts 用于导入各种模块。

app-routing.module.ts 用于规定本 Angular 项目的路由信息，规定了路由跳转的规则，具体内容如下：

```
let routes: Routes = [];

if (localStorage.getItem( key: 'teacher')) {
  routes = [
    {path: '', redirectTo: 'main', pathMatch: 'full'},
    {path: 'login', component: MainComponent},
    {path: 'main', component: MainComponent},
    {path: 'detail', component: LessontabComponent},
    {path: 'register', component: RegistertableComponent}
  ];
} else {
  routes = [
    {path: '', redirectTo: 'login', pathMatch: 'full'},
    {path: 'login', component: LoginComponent},
    {path: 'main', component: LoginComponent},
    {path: 'detail', component: LoginComponent},
    {path: 'register', component: RegistertableComponent}
  ];
}

@NgModule({
  declarations: [],
  imports: [
    CommonModule,
    RouterModule.forRoot(routes)
  ],
  exports: [RouterModule]
})

export class AppRoutingModule {
}
```

其他的文件由于数量略多并且与本 PJ 关系不是很大，所以在此略去。

二、 教师端关键功能的实现细节:

1. 发布课程:

关键代码:

```
export class LessontableComponent implements OnInit {
  @ViewChild(MatHorizontalStepper) Stepper: MatHorizontalStepper;
  isLinear = true;
  firstFormGroup: FormGroup;
  secondFormGroup: FormGroup;
  thirdFormGroup: FormGroup;
  fourthFormGroup: FormGroup;

  constructor(private _formBuilder: FormBuilder, private service: LessonService) {
  }

  ngOnInit() {
    this.firstFormGroup = this._formBuilder.group( controlsConfig: {
      firstCtrl: ['', Validators.required]
    });
    this.secondFormGroup = this._formBuilder.group( controlsConfig: {
      secondCtrl: ['', Validators.required]
    });
    this.thirdFormGroup = this._formBuilder.group( controlsConfig: {
      thirdCtrl: ['', Validators.required]
    });
    this.fourthFormGroup = this._formBuilder.group( controlsConfig: {
      fourthCtrl: ['', Validators.required]
    });
    console.log(this.Stepper);
  }

  add() {
    var formdata = new FormData();
    var teacher = JSON.parse(localStorage.getItem( key: 'teacher'));
    const objFile = document.getElementById( elementId: 'img') as HTMLInputElement;
    formdata.append( name: 'teacher_id', teacher.id);
    formdata.append( name: 'title', this.firstFormGroup.get('firstCtrl').value);
    formdata.append( name: 'subtitle', this.secondFormGroup.get('secondCtrl').value);
    formdata.append( name: 'background', objFile.files[0]);
    formdata.append( name: 'introduce', this.fourthFormGroup.get('fourthCtrl').value);
    formdata.append( name: 'content', value: '[]');
    this.service.addLesson(formdata).subscribe(data => {
      location.reload();
    });
  }
}
```

前端页面用的是 angular material 中的步进器，ngOnInit 中为每个表单项安装了验证器，用于验证输入是否合法，add 方法获取所有表单数据的输入并将数据写入 Formdata 对象中，通过 service 提交给后台，此时的课程只包括课程名、介绍等信息，课程内容的编写在另一个模块中。

2. 发布课程内容:

a、 章节安排:

```
openDialog2(): void {
  const dialogRef = this.dialog.open(ChapterdialogComponent, config: {
    width: '250px',
    data: {name: this.name, animal: this.animal}
  });

  dialogRef.afterClosed().subscribe( next: result => {
    if (result) {
      this.lesson = JSON.parse(localStorage.getItem( key: 'lesson'));
      console.log('The dialog was closed');
      this.animal = result;
      console.log(this.animal);
      temp.chapter_name = this.animal;
      temp.section = [];
      this.lesson.chapters.push(temp);
      localStorage.setItem('lesson', JSON.stringify(this.lesson));
    }
  });
}

upChapter(i: number) {
  this.lesson = JSON.parse(localStorage.getItem( key: 'lesson'));
  if (i > 0) {
    temp = this.lesson.chapters[i - 1];
    this.lesson.chapters[i - 1] = this.lesson.chapters[i];
    this.lesson.chapters[i] = temp;
  }
  localStorage.setItem('lesson', JSON.stringify(this.lesson));
  if (this.ChaIndex == i) {
    this.ChaIndex = i - 1;
    this.changeIndex(this.SecIndex, this.ChaIndex);
  }
}

downChapter(i: number) {
  this.lesson = JSON.parse(localStorage.getItem( key: 'lesson'));
  if (i < this.lesson.chapters.length - 1) {
    temp = this.lesson.chapters[i + 1];
    this.lesson.chapters[i + 1] = this.lesson.chapters[i];
    this.lesson.chapters[i] = temp;
  }
  localStorage.setItem('lesson', JSON.stringify(this.lesson));
  if (this.ChaIndex == i) {
```

由于该模块内容较多，此处只截取一部分，通过对 lesson 对象的章节列表的修改实现章节的编辑，以 opendialog 函数为例：

通过一个打开的对话框获取用户的所设置的新的章的名字，在对话框关闭后，若用户设置了章的名字，则通过 localstorage 获取当前课程的数据结构，并将新写入的章加入到章列表中，然后再将改变后的课程写回 localStorage 中。章节的移动、删除、更名等操作的大体逻辑与此相似。

b、　增加课程内容：

```
addQuestion() {
  this.lesson = JSON.parse(localStorage.getItem( key: 'lesson'));
  qa.question = '';
  qa.answer = '';
  this.lesson.chapters[this.ChaIndex].section[this.SecIndex].question.push(qa);
  localStorage.setItem('lesson', JSON.stringify(this.lesson));
}

tempSave() {
  localStorage.setItem('lesson', JSON.stringify(this.lesson));
}

saveQA() {
  localStorage.setItem('lesson', JSON.stringify(this.lesson));
  this.service.updateLesson(this.lesson);
  alert('已保存');
}

upQA(i: number) {
  if (i > 0) {
    qa = this.lesson.chapters[this.ChaIndex].section[this.SecIndex].question[i - 1];
    // tslint:disable-next-line:max-line-length
    this.lesson.chapters[this.ChaIndex].section[this.SecIndex].question[i - 1] =
      this.lesson.chapters[this.ChaIndex].section[this.SecIndex].question[i];
    this.lesson.chapters[this.ChaIndex].section[this.SecIndex].question[i] = qa;
    localStorage.setItem('lesson', JSON.stringify(this.lesson));
  }
}
```

课程的内容是设置问题及相应的答案，通过 addquestion 添加问题，也是从 LocalStorage 中获取 lesson 数据，将内容写入，再将修改后的

数据写入，同时也实现了内容的移动及删除功能。

c、 增加作业：

```
addSC() {
  this.lesson = JSON.parse(localStorage.getItem( key: 'lesson'));
  SC.question = '';
  SC.choiceA = '';
  SC.choiceB = '';
  SC.choiceC = '';
  SC.choiceD = '';
  SC.right_choice = 'A';
  this.lesson.chapters[this.ChaIndex].section[this.SecIndex].singleChoice.push(SC);
  localStorage.setItem('lesson', JSON.stringify(this.lesson));
}

upSC(i: number) {
  if (i > 0) {
    SC = this.lesson.chapters[this.ChaIndex].section[this.SecIndex].singleChoice[i - 1];
    // tslint:disable-next-line:max-line-length
    this.lesson.chapters[this.ChaIndex].section[this.SecIndex].singleChoice[i - 1] =
      this.lesson.chapters[this.ChaIndex].section[this.SecIndex].singleChoice[i];
    this.lesson.chapters[this.ChaIndex].section[this.SecIndex].singleChoice[i] = SC;
    localStorage.setItem('lesson', JSON.stringify(this.lesson));
  }
}

downSC(i: number) {
  if (i < this.lesson.chapters[this.ChaIndex].section[this.SecIndex].singleChoice.length - 1) {
    SC = this.lesson.chapters[this.ChaIndex].section[this.SecIndex].singleChoice[i + 1];
    // tslint:disable-next-line:max-line-length
    this.lesson.chapters[this.ChaIndex].section[this.SecIndex].singleChoice[i + 1] =
      this.lesson.chapters[this.ChaIndex].section[this.SecIndex].singleChoice[i];
    this.lesson.chapters[this.ChaIndex].section[this.SecIndex].singleChoice[i] = SC;
    localStorage.setItem('lesson', JSON.stringify(this.lesson));
  }
}
```

课程的作业主要是有 ABCD 四个选项的单选题，addSC 方法通过设置方法的问题、四个选项及正确选项的信息，并将其写入课程数据对象中来实现作业的增添，同时也实现了对作业的移动、删除等功能。

3. 查看学生选课情况：

```
export interface PeriodicElement {
  student_id: string;
  name: string;
  studentnumber: string;
  progress: number;
}

//let temp = {} as PeriodicElement;
let ELEMENT_DATA: PeriodicElement[] = [];
let courseStudent = {} as Course_student;
let Stu = {} as Student;

@Component({
  selector: 'app-studenttable',
  templateUrl: './studenttable.component.html',
  styleUrls: ['./studenttable.component.css']
})
export class StudenttableComponent implements OnInit {
  @ViewChild(MatTable) private table: MatTable<any>;
  displayedColumns: string[] = ['student_id', 'name', 'studentnumber', 'progress'];
  dataSource: PeriodicElement[] = [];
  lesson: Course;
  op: number;

  constructor(private service: StudentService) {
  }

  ngOnInit() {
    this.dataSource = [];
    this.op = 0;
    this.lesson = JSON.parse(localStorage.getItem( key: 'lesson'));
    this.service.getStudnetList(this.lesson.id).subscribe(data => {
        if (data.length !== 0) {
          this.dataSource = data;
        }
      }
    );
  }
```

前端利用了 angular 的 material 包中的 table 模块，通过填充 dataSource 数据结构实现在前端的展示，ngOnInit 方法通过从后台拉取数据并填入 dataSource 中实现对于选课的学生的信息的显示，显示的信息包括

学生 id、名字、学号以及进度。

4. 查看学生提交作业情况：

```
ngOnInit() {
  var request = {
    course_id: this.data.lesson.id,
    chapter: this.data.lesson.chapters[this.data.ChaIndex].chapter_name,
    section: this.data.lesson.chapters[this.data.ChaIndex].section[this.data.SecIndex].sectionname
  };
  this.service.getAnswerRecord(request).subscribe(data => {
    var answers = data;
    for (let i = 0; i < this.data.lesson.chapters[this.data.ChaIndex].section[this.data.SecIndex].singleChoice.length; i++) {
      ELEMENT_DATA = [];
      var question = this.data.lesson.chapters[this.data.ChaIndex].section[this.data.SecIndex].singleChoice[i].question;
      for (const ar of answers) {
        if (ar.question == question) {
          let temp2 = {
            id: ar.id,
            name: ar.name,
            choice: ar.choice,
            right_choice: this.data.lesson.chapters[this.data.ChaIndex].section[this.data.SecIndex].singleChoice[i].right_choice
          };
          ELEMENT_DATA.push(temp2);
        }
      }
      this.dataSource.push(ELEMENT_DATA);
    }
    this.tables.forEach(table => table.renderRows());
  });
}
onNoClick(): void {
```

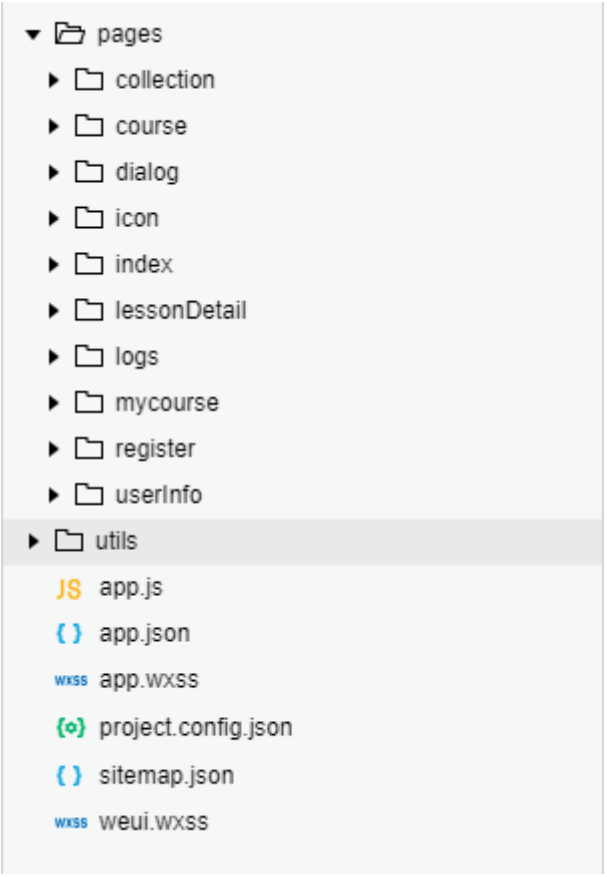
前端显示提交作业情况的也是一个 table,显示的内容为 dataSource。

ngOnInit 方法从后台读取提交作业的情况，并写入 dataSource 数据对象中，最后在前端进行展示。

三、 教师端部署方法及地址：

教师端通过 nginx 部署在 EC2 上，地址为 http://54.90.160.124，由于 js 文件较大并且服务器在国外，第一次打开需要较长时间。

四、 学生端项目结构：



Page 目录下是各个页面的文件，包括 js、json、wxml、wxss 等文件。

Collection 是用户的收藏页面，course 课程展示及搜索页面，dialog 是课程的对话页面，icon 是项目用到的一些图标，index 是微信的授权页面，lessonDetail 是课程的章节列表，logs 用于生成日志文件，mycourse 展示用户选择的课程，register 则是注册页面，userInfo 展示用户个人的一些信息并提供修改（如性别、学号等）。

在其他的文件中，App.js 是全局的一个 JS 文件，app.json 用于对项目整体进行配置，app.wxss 是全局的 css 样式，weui 是 WEUI 样式库，被导入到 app.wxss 中。

五、 学生端关键功能的实现细节：

1. 完善个人信息（注册）：

```
let nickname = wx.getStorageSync('username');
var that = this;
var newStudent = {
  id:this.uuid(),
  signature:wx.getStorageSync('openid'),
  mailbox:this.data.information.phone,
  sex:this.data.userSex,
  name:this.data.information.name,
  studentnumber:this.data.information.idNumber
}
wx.request({
  url: 'http://52.91.208.255:8080/addstu',
  method: 'Post',
  data: newStudent,
  headers: {
    'Content-Type': 'json'
  },
  success: function (res) {
    wx.request({
      url: 'http://52.91.208.255:8080/student?signature=' + wx.getStorageSync('openid'),
      headers: {
    'Content-Type': 'json'
  },
  success:function(res){
    wx.setStorageSync('uid', res.data.id)
  }
    })
    wx.setStorageSync('userID', newStudent.id);
    wx.showToast({
      title: '提交成功',
      icon: 'success'
    })
    that.setData({
      modalHidden: true
    })
    wx.switchTab({
      url: '/pages/course/course'
    })
  }
```

前端页面是一个表单。通过获取表单上的输入来构造 newStudent 对象，并在 requset 中把这个对象发给后台以实现注册，若注册成功，则进行相应的提示并跳转到主页面。

2. 查看并选择课程：

   查看课程：

```
wx.request({
  url: 'http://52.91.208.255:8080/course/getallcourses',
  headers: {
    'Content-Type': 'application/json'
  },
  success: function (res) {
    console.log(res);
    that.setData({
      lesson: res.data
    })
    //let uid = wx.getStorageSync('userID');
    wx.request({
      url: 'http://52.91.208.255:8080/course/getcoursebyid?id=' + wx.getStorageSync('uid'),
      headers: {
        'Content-Type': 'application/json'
      },
      success: function (res) {
        console.log(res.data)
        that.setData({
          inStudy: res.data
        })
        that.data.canStudy = [];
        for (var i = 0; i < that.data.lesson.length; i++) {
          if (that.data.inStudy.length > 0) {
            for (let j = 0; j < that.data.inStudy.length; j++) {
              if (that.data.inStudy.indexOf(that.data.lesson[i].id) < 0) {
                that.data.canStudy.push(1);
              } else {
                that.data.canStudy.push(0);
              }
            }
          } else {
            that.data.canStudy.push(1);
          }
        }
        that.setData({
          canStudy: that.data.canStudy
        });
        console.log(that.data.canStudy);
        // console.log(that.data.inStudy.indexOf('ddd'));
      }
```

通过从后台获取所有的课程，并根据是否已经选择这门课程，显示该课程是否可以进行选课。

选择课程：

选择课程主要是通过 addCourse 实现的。

```
addCourse(e) {
  var id = e.currentTarget.dataset.pp;
  console.log(id)
  var that = this;
  let uid = wx.getStorageSync('uid');
  var temp = {
    id: 1,
    course_id: id,
    student_id: uid,
    progress: 0,
    studysection: ''
  }
  wx.request({
    url: 'http://52.91.208.255:8080/course/addcourse',
    method: 'post',
    data: temp,
    headers: {
      'Content-Type': 'application/json'
    },
    success: function(res) {
      for (let i = 0; i < that.data.lesson.length; i++) {
        if (that.data.lesson[i].id == id) {
          that.data.canStudy[i] = 0;
          that.setData({
            canStudy: that.data.canStudy
          })
          console.log('ooo')
          break;
        }
      }

    }
  })
},
```

用户点击了选课的按钮后，后台进行选课记录的填写。通过填写 temp 数据的内容并发往后台进行记录保存来实现选课。

3. 按照对话式教学学习课程：

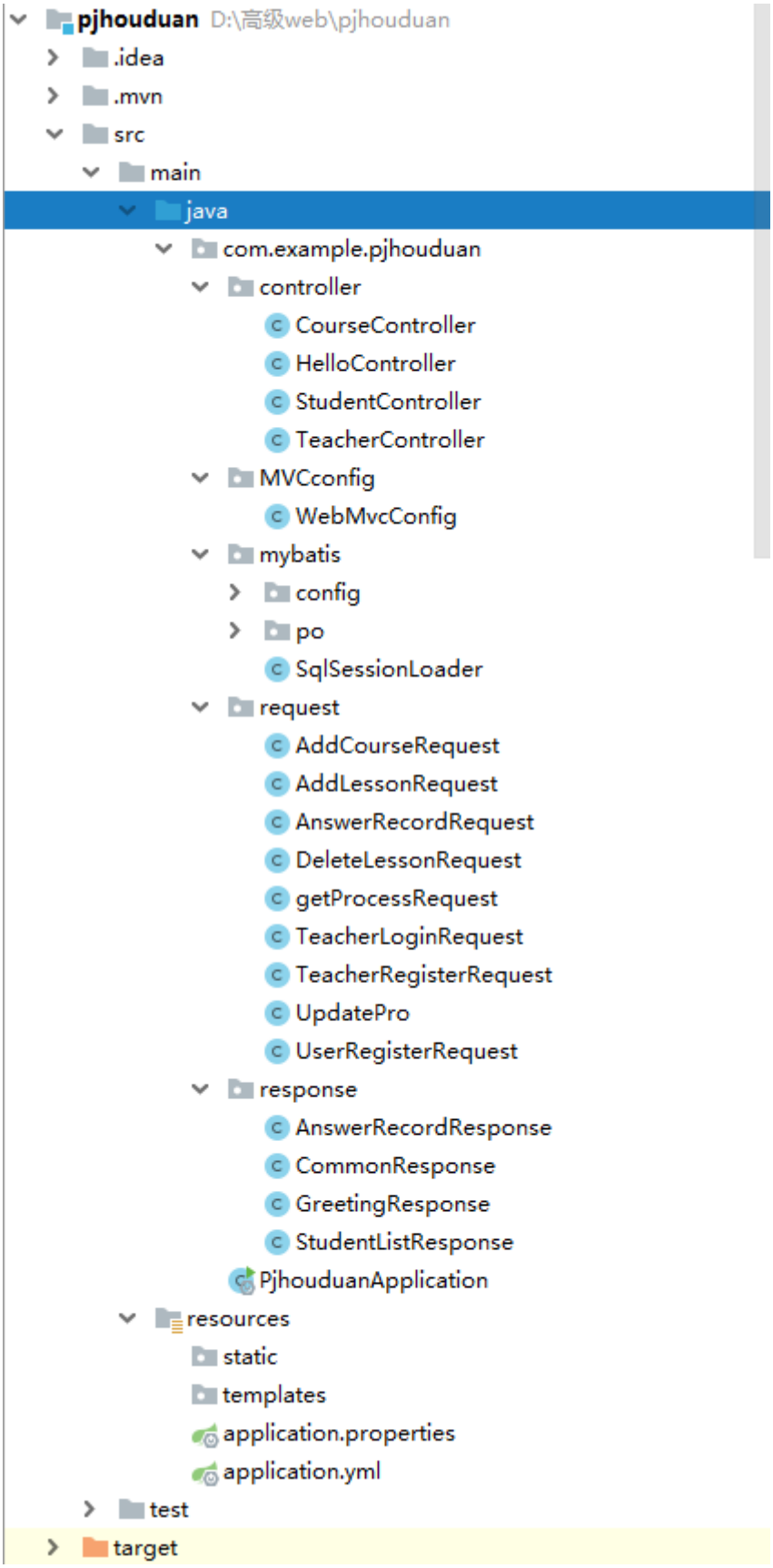
```
addContent(e) {
  if (this.data.QAIndex < this.data.lesson.chapters[this.data.ChaIndex].section[this.data.SecIndex].question.length) {
    msgList.push({
      speaker: 'server',
      contentType: 'text',
      content: this.data.lesson.chapters[this.data.ChaIndex].section[this.data.SecIndex].question[this.data.QAIndex].question,
      canShou:true
    })
    msgList.push({
      speaker: 'costomer',
      contentType: 'text',
      content: this.data.lesson.chapters[this.data.ChaIndex].section[this.data.SecIndex].question[this.data.QAIndex].answer
    })
    this.data.QAIndex++;
    this.setData({
      msgList,
      QAIndex: this.data.QAIndex
    });
  } else {
    msgList.push({
      speaker: 'server',
      contentType: 'text',
      content: '你已学完所有内容，下面进入作业阶段'
    })
    this.setData({
      msgList,
      hasListAll: true
    });
    this.makeSingle();
  }
```

AddContent 通过获取的课程内容，并通过将其推入 msgList 中进行显示。

六、 PJ 后端项目结构;

由于学生端和教师端的关键功能的大部分细节已经进行了展示，而在后端更多的只是进行与数据库和本地文件系统的交互，故不展示后台的具体的代码实现，仅对后台 PJ 整体的项目结构进行介绍。

具体的项目结构图在下一页：



Src 目录下的 java 文件夹是后端的主体内容。

Controller 文件夹下是各种控制器，用于响应前台的请求，例如TeacherController 用于响应教师端的一些请求。

MVCconfig 文件夹用于实现本地存储路径与网络路径的映射，用于回显图片。

Mybatis 文件夹用于存放与 Mybatis 相关的一些文件。Config 文件夹对mybatis 进行配置，po 文件夹则包括 mybatis 与数据库进行交互时用到的数据原型，SqlsessionLoader 则包装了 Mybatis 的工厂方法，用于生成Mybatis 的 sql 事务。

Request 文件夹则定义了一些后端接收的请求体的格式，其下的 java 文件均包含一个无参的构造函数。

Response 文件夹则定义了后端放回的数据相应的格式，前端接收该相应体并进行相应处理。

PjhouduanApplication 是后端的启动文件。

Resources 文件夹用于存放 PJ 的一些静态资源，如模板（本后端没有），配置等。Application.properties 以及 application.yml 均是后端的配置文件。

Test 文件夹下是一些测试文件。

Target 文件夹则是一些 SpringBoot 导出的文件。

七、 服务器的部署配置：

服务器通过两个 Docker 容器进行配置。其中一个 Docker 容器运行 SpringBoot 导出的 jar 包，并映射在服务器 80 端口，另一个容器则是一个 mysql 容器，用于存放 PJ 的数据，映射在服务器的 3306 端口。这两个

容器均部署在一台 ubuntu 服务器上。服务器的地址是 http://52.91.208.255:8080。