勘误：

1. 封装模块中需要导入组件依赖模块
2. 不要在sql中拼参数

知识扩展

1. 组件通信
    - 父组件向子组件通信
        - 输入属性`@Input() uname`

          ** 使用场景：不关心属性变化，没有额外操作
        - Setter拦截器

```
private _uname: string;

get uname(): string {
  return this._uname;
}

@Input()
set uname(value: string) {
  this._uname = (value && value.trim()) || '无名氏';
}
```

          ** 使用场景：希望加工输入属性的值
        - ngOnChanges拦截

```
export class ChildComponent implements OnInit, OnChanges {
  ngOnChanges(changes: SimpleChanges): void {
    // 当前组件中成员属性值发生变化时，我有额外的事情要做时
    const chng = changes['uname'];
    let log: string;

    if (chng) { // 确定变化发生在uname上
      // 获取当前值和之前值，日志记录
      const currValue = chng.currentValue;
      // 是否首次赋值
      if (chng.isFirstChange()) {
        log = `设置uname初始值为${currValue}`;
      } else {
        const previousValue = chng.previousValue;
        log = `uname值由${previousValue}为${currValue}`;
      }
      this.changeLog.push(log);
    }
  }

}
```

** 使用场景：当数值发生变化后要做额外操作

- 子组件向父组件
  - 输出属性

```
@Output()
change: EventEmitter<string> = new EventEmitter<string>();
changeName() {
    this.uname = 'BlaBla';
    // 通知父组件
    this.change.emit(this.uname);
}
```

  - 模板引用变量

```
<app-child [uname]="uname" #child></app-child>
<p>{{child.uname}}</p>
```

  - @ViewChild

```
export class CompCommunicateComponent implements OnInit, AfterViewInit {
    @ViewChild('child') child: ElementRef;
    ngAfterViewInit(): void {
        // console.log(this.child);
        console.log(this.child['uname']);
    }
}
```

- 兄弟组件 Subject
  - 创建服务

```
export class CommunicateService {
  // 发送数据源
  private subject = new Subject<string>();
  // 消费者用来监听的Observable
  ob = this.subject.asObservable();

  emit(msg) {
    this.subject.next(msg);
  }

}
```

  - 组件1中发送数据

```
this.cs.emit('大哥，我改名了！！');
```

- 组件2中消费数据

```
this.cs.ob.subscribe(
  msg => alert(msg)
);
```

项目：

1. 上传头像：
   - 安装模块：npm i -S ngx-uploader
   - 导入模块：`imports: [NgxUploaderModule]`
   - 添加上传指令：

```
<input type="file" ngFileSelect
       (uploadOutput)="onUploadOutput($event)"
       [uploadInput]="uploadInput">
```

   - 执行上传和响应处理

```
uploadInput: EventEmitter<UploadInput> = new EventEmitter<UploadInput>();
onUploadOutput(output: UploadOutput) {
  if (output.type === 'allAddedToQueue') {
    // 开始上传
    this.uploadInput.emit({
      type: 'uploadAll',
      url: '/api/users/uploadAvatar',
      method: 'POST'
    });
  } else if (output.type === 'done') {
    // 上传完成
    if (output.file.responseStatus === 200 && output.file.response.success) {
      this.avatar = avatarSrc + output.file.response.data;
    } else {
      alert('上传失败! ');
    }
  }
}
```

2. 课程列表
   - 容器型组件：

```
<div class="kkb-panel-body">
  <ng-content></ng-content>
</div>
```

   - 关联查询：

```
select c.id,c.name,c.phase,vc.poster from user_clazz uc
    left join clazz c on uc.clazz_id = c.id
    left join vip_course vc on c.course_id = vc.id
    where user_id=?
```