

# 成员分工与个人总结

## 一、成员分工及比例

分工	成员
微信小程序端	董达芬 25%
	符丽雪 25%
教师端	卢振洁（组长） 25%
	闫世艺 25%

## 二、个人总结

### （1）董达芬

#### I、遇到的问题及解决方法

##### 1、mpvue 获取后端数据：

问题：mpvue 获取 springboot 提供的接口的数据与小程序获取的方式不一样，像 lab2 一样封装了数据但是用 mpvue 还是获取不到数据，尝试用 axios 发送 ajax 请求会报各种奇怪的 bug。

解决方法：使用 flyio 请求数据。

（1）先安装 flyio，终端命令行：npm install flyio --save.

（2）引入 request.js，见 utils 目录，将此文件引入 main.js 然后直接挂载到 vue 原型上

```
import fly from '../utils/request'
import Vue from 'vue'
```

（3）然后在页面或者组件中直接使用

```
this.$fly.get('http://localhost:8084/home/listAllCourse').then(res => {
  this.courses = res;
```

##### 2、请求数据跨域

问题：前端请求不到后端数据，因为为了实现后端的实现不允许跨域请求。

解决方法：在根目录下引入 CorsConfig 类，允许跨域请求。

```

public class CorsConfig {
    private CorsConfiguration buildConfig() {
        CorsConfiguration corsConfiguration = new CorsConfiguration();
        corsConfiguration.addAllowedOrigin("*"); // 1允许任何域名使用
        corsConfiguration.addAllowedHeader("*"); // 2允许任何头
        corsConfiguration.addAllowedMethod("*"); // 3允许任何方法 (post、get等)
        return corsConfiguration;
    }
}

```

### 3、端口占用

问题：后端 springboot 和 mpvue 前端都默认监听 8080 端口，造成端口占用问题。

解决方法：springboot 的端口比较容易修改，所以在 application.properties 中设置

```
server.port=8084
```

### 4、找不到 build 方法

```

InputStream inputStream = Resources.getResourceAsStream("SqlMapConfig.xml");
sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);

```

Cannot resolve method 'build(java.io.InputStream)'

解决方法：网上查到的要将 reader 和 builder 方法分开写。

```

String resource = "SqlMapConfig.xml";
Reader reader = Resources.getResourceAsReader(resource);
SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();
sqlSessionFactory = builder.build(reader);

```

## II、个人负责部分

### Springboot 后端：

- Controller 目录

(1) CourseController 目录：获取所有课程信息，用于小程序首页；实现根据输入搜索框的课程名查询相应的课程信息；根据用户在首页点击的课程获取该课程的详细信息。

(2) MessageController 目录：获取消息记录；发送消息（将消息插入数据库）。

- responses 目录：

(1) 实现了 SuccessResponse 类

(2) 实现了 ErrorResponse 类

- mapper 目录：根据上述的功能实现相应的 SQL 语句。

### 小程序前端

- 实现了 pages/index 所有课程信息的显示


- pages/test 指定课程详情界面

- pages/chat 聊天页面信息显示

### (2)、符丽雪

## I、遇到的问题及解决方法

- 监听文本框输入数据的变化，进而获取文本框中的数据。添加 v-model

```
placeholder="搜索课程" v-model="titleSearch" v-on:input="getTitle" />
 @click="searchCourse(titleSearch)" />
```

```
//监听文本框的数据变化
getTitle(){
  if(this.titleSearch!==''){
    console.log(this.titleSearch)
  }
},
```

- 搜索当前学生登录状态下，能够添加收藏课程和已参与课程。大致思路就是获取当前学生的昵称，再与数据库匹配（我们默认昵称是不同的）得到相应的 ID，所以根据相应的 ID 和相应 button 的功能分别插入对应的收藏和已参与表。

## II、个人负责部分

### 后端 springboot 的实现部分

#### Controller 目录：

- StudentController 类  
实现了按昵称查找学生（findStudentByNick），路由：/stu/findStuByNickName。  
实现了根据学生填的表单注册学生或更新学生的信息，路由：/stu/msg。
- CourseController 类  
实现了添加收藏课程、查询收藏课程  
实现了添加参与课程、查询参与课程
- Domain 目录下的所有实体类

#### request 目录：

- 实现了 SqlSessionLoader

#### response 目录：

- 实现了 CourseResponse 类
- 实现了 StudentCollectResponse 类
- 实现了 AddCourseResponse 类

#### mapper 目录：

- 根据上述的函数（功能）实现相应的 SQL 语句。

### 前端（小程序端）的实现部分

- 实现了（pages/login、pages/userInfo 和 pages/courseMsg）用户个人信息的更新/注册以及收藏功能（点击按钮后跳转页面查看收藏课程）
- 实现了（pages/index）中的搜索框搜索功能（但是数据没显示出来）
- 实现了（pages/takeCourse）中点击按钮查看已参与课程

### （3）卢振洁

## I、遇到的问题及解决方法

- 跨域：

问题：前端页面端口为 63342，后端请求的端口为 8080，前端请求不到后端

解决方法：在 controller 代码前加上注解 **@CrossOrigin**

原理：跨源资源共享（CORS）是由大多数浏览器实现的 W3C 规范，允许您灵活地指定什么样的跨域请求被授权，而不是使用一些不太安全 and 不太强大的策略，如 IFRAME 或 JSONP。注解 **@CrossOrigin** 用于启用 CORS

- 页面间传参:

问题: 当点击某个页面课程进入课程详情时需要传递课程 id 给下一个页面。

解决方法: 利用 **url 传参**, 利用字符串拼接将要传的数据放进 url 并在下一个页面进行分割以便获取。

传入: `:href="'blogCourse.html?courseID='+course.courseID"`

```
//从url里获取当前课程id
function getCourseID() {
    var url = decodeURI(window.location.href);
    var argsIndex = url.split("?courseID=");
    var arg = argsIndex[1];
    return arg;
}
```

获取:

- 利用 **cookie 存取** 数据:

问题: 用户的 id 一旦登录进去以后就不会改变, 且会被多个页面共同使用, 这个时候利用 url 传参会更加复杂, 所以我们使用了 cookie 来存取, 非常简单方便。

存入: `$.cookie('teacherID',result.teacherID);`

取出: `var teacherID = $.cookie('teacherID');`

## II、个人负责部分

- 课程相关页面: 首页, 课程详情, 我的课程

具体文件: **html**:blog.html/blogCourse.blog/myCourse.html

**Js**:blog.js/blogCourse.js/myCourse.js

- 课程相关服务: 查询课程、开设课程、更改课程相关内容

具体文件: **controller**:CourseController

**Request**:chapterRequest/CourseRequest/HomeworkRequest  
/Teacherrequest

**Mapper**:CourseMapper.xml

### (4) 闫世艺

#### I、遇到的问题及解决方法

- 关于数据库的设计:

编码前对数据库的设计不可能完全符合预期, 比如字段的类型, 表之间的连接等。在编码实现阶段会发现当前数据库不符合预期的情况, 然后再对数据库进行修改, 基本上重新建表。所以如果前期设计不够完善的话, 后期实现的时候谁会出现一些问题的, 也会花一些时间在更新数据库上面。

#### II、个人负责部分

- 数据库: 数据库的设计, 创建, 和部署

- 实现:

- 课程相关页面: 登录, 注册, 开设新的课程, 个人信息

具体文件: **html**: index.html/register.html/createNewCourse/html/userInfo.html

**Js**: index.js/register.js /userInfo.js

- 课程相关服务: 教师登录和注册的认证, 以及个人信息的展示和修改

具体文件: **Controller**: TeacherController

Request: TeacherInfo/TeacherLogin/TeacherRegister/  
TeacherModify/TeacherRequest

Response: MessageResponse/LoginResponse/RegisterResponse

Mapper: Teacher.xml

- 文档: 项目设计文档后端部分, 数据库设计, 文档整理