[TOC]

## 前后端交互设计

### 概述

后端API是在RESTful风格的,发送和返回内容均为json数据。

若操作成功：

* GET 直接返回获取的内容
* POST和PUT 返回修改或创建的条目的ID，以及成功消息
* DELETE 返回成功消息

正常返回数据：  
{  
 code : '0', // '0' 代表成功  
 data : {data} // 按实际要求封装  
}

无返回数据：  
{  
 code : '0', // '0' 代表成功  
 data : null  
}

异常返回：  
{  
 code : String // 非 '0' 时表示异常代码  
 msg : String // 具体的错误信息  
}

### 数据

#### 学生用户登录注册相关

1001 注册  
POST /register  
Params {  
 registerName : String （用户名）  
 registerPass : String (加密后的密码)  
}  
Response {  
 code : '0'  
}  
  
Response Excetion ： { code : '', msg : ''}  
code msg  
1001A 登录名已存在

1002 登录并获取统计数据  
POST /login  
Params {  
 loginName : String （用户名）  
 loginPass : String （加密后的密码）  
}  
Response {  
 code : '0',  
 data : {  
 statistics : { // 统计数据  
 howMuch : Number, // 已经交了多少班费  
 howMuchRemain : Number, // 还差多少要交  
 numOfDayNotSign : Number, // 旷课总数  
 dayNotSign : [Date] // 旷课情况  
 }  
 }  
}  
  
Response Excetion ： {code : '', msg : ''}  
code msg  
1002A 用户不存在  
1002B 密码不正确

1003 登出  
GET /logout  
Params  
 null  
Response {  
 code : '0'  
}  
  
Response Excetion ： {code : '', msg : ''}  
code msg  
1003A 未知错误

#### 管理员相关

2001 获取所有学生  
GET /manager/students  
Params  
 null  
Response {  
 code : '0',  
 data : {  
 students : [{  
 authorId : ObjectId, // 学生的流水号，也是学号Id  
 roleId : Number // 学生的roleId  
 }]  
 }  
}  
  
Response Excetion ： {code : '', msg : ''}  
code msg  
2001A 数据库查询错误

2002 更新某个学生的权限  
PUT /manager/update  
Params {  
 student : {  
 authorId : ObjectId, // 学生的流水号，也是学号Id  
 roleId : Number // 学生的新roleId  
 }  
}  
Response  
{  
 code : '0'  
}  
  
Response Excetion ： {code : '', msg : ''}  
code msg  
2002A 找不到这个学生  
2002B 没有这种权限等级

#### 帖子相关

3001 获取某个班级全部开班会帖  
GET /meeting  
Params {  
 classId : String // 哪个班  
}  
Response {  
 code : '0',  
 data : {  
 meetings : [{  
 classId : String, // 班级ID  
 date : Date, // 开会时间  
 place : String, // 开会地点  
 title : String, // 主题  
 num : Number, // 应到人数  
 gotten : Number, // 实到人数  
 pAgree : [Number], // 同意的人的ID  
 pDisagree : [Number], // 不同意的人的ID  
 authorId : Number // 发帖人ID  
 }]  
 }  
}  
  
Response Excetion ： {code : '', msg : ''}  
code msg  
3001A 数据库查询错误

3002 发表开班会帖  
POST /meeting  
Params {  
 meeting : {  
 classId : String, // 班级ID  
 date : Date, // 开会时间  
 place : String, // 开会地点  
 title : String, // 主题  
 num : Number, // 应到人数  
 gotten : Number, // 实到人数  
 pAgree : [Number], // 同意的人的ID  
 pDisagree : [Number], // 不同意的人的ID  
 authorId : Number // 发帖人ID  
 }  
}  
Response {  
 code : '0'  
}  
  
Response Excetion ： {code : '', msg : ''}  
code msg  
3002A 数据库查询错误

3003 获取全部收班费帖  
GET /classMoney  
Params {  
 classId : String // 哪个班  
}  
Response  
{  
 code : '0',  
 data : {  
 classMoneys : [{  
 classId : String, // 班级ID  
 date : Date, // 截止时间  
 payedMembers : [Number], // 已经缴费的人的ID  
 unpayedMembers : [Number], // 尚未缴费的人的ID  
 authorId : Number // 发帖人ID  
 }]  
 }  
}  
  
Response Excetion ： {code : '', msg : ''}  
code msg  
3003A 数据库查询错误

3004 发表收班费帖  
POST /classMoney  
Params {  
 classMoneys : {  
 classId : String, // 班级ID  
 date : Date, // 截止时间  
 payedMembers : [Number], // 已经缴费的人的ID  
 unpayedMembers : [Number], // 尚未缴费的人的ID  
 authorId : Number // 发帖人ID  
 }  
}  
Response {  
 code : '0'  
}  
  
Response Excetion ： {code : '', msg : ''}  
code msg  
3004A 数据库查询错误

3005 获取评优等生帖  
GET /goodStudent  
Params {  
 classId : String // 哪个班  
}  
Response  
{  
 code : '0',  
 data : {  
 goodStudents : [{  
 classId : String, // 班级ID  
 title : String, // 奖项名称  
 stuId : Number, // 获奖人ID  
 bond : Number, // 奖金  
 authorId : Number // 发帖人ID  
 }]  
 }  
}  
  
Response Excetion ： {code : '', msg : ''}  
code msg  
3005A 数据库查询错误

3006 发表评优等生帖  
POST /goodStudent  
Params {  
 goodStudents : [{  
 classId : String, // 班级ID  
 title : String, // 奖项名称  
 stuId : Number, // 获奖人ID  
 bond : Number, // 奖金  
 authorId : Number // 发帖人ID  
 }]  
}  
Response {  
 code : '0'  
}  
  
Response Excetion ： {code : '', msg : ''}  
code msg  
3004A 数据库查询错误

#### 关于其他发布/获取全部帖子接口

发布考勤结果帖接口结构与其他帖子的类似，不再赘述

## 关于一些其他问题的设计说明

### HTTP redirect该使用何种status code，各有何种优劣性

常用的重定向方式有：301 redirect、302 redirect 与meta fresh。

* 301 redirect

301代表永久性转移，301重定向是网页更改地址后对搜索引擎友好的最好方法，只要不是暂时搬移的情况，都建议使用301来做转址。

* 302 redirect

302代表暂时性转移，在前些年，不少Black Hat SEO（黑帽SEO）曾广泛应用这项技术作弊，目前，各大主要搜索引擎均加强了打击力度，像Google前些年对域名之王（Business）以及近来对BMW德国网站的惩罚。即使网站客观上不是spam，也很容易被搜寻引擎容易误判为spam而遭到惩罚。

* meta fresh

这在2000年前比较流行，不过现在已很少见。其具体是通过网页中的meta指令，在特定时间后重定向到新的网页，如果延迟的时间太短(约5秒之内)，会被判断为spam。

另外还有另外一些情况

* 302（Found）

请求的资源现在临时从不同的URI响应请求。除非指定了Cache-Control或Expires，否则该响应不可缓存。 如果当前请求非HEAD或GET，浏览器需取得用户确认，再进行重定向。这很好理解，因为上下文发生了变化，比如POST请求不是幂等的。

* 303（See Other）

对应当前请求的响应可以在另一个URI上被找到，而且客户端应当采用GET的方式访问那个资源。 这个方法的存在主要是为了允许由脚本激活的POST请求输出重定向到一个新的资源。 303响应禁止被缓存。303会使得浏览器直接GET那个资源，不需用户同意。这是Web应用中最常见的重定向方式。

* 304（Not Modified）

如果客户端发送了一个带条件的GET请求且该请求已被允许，而文档的内容（自上次访问以来或者根据请求的条件）并没有改变。 304响应禁止包含消息体。

304响应也是一种缓存机制。Web服务器对静态资源文件通常会采取缓存，因此在Web开发中你可以看到大量的304响应。 服务器给出的相应中通常会包含Etag来标识资源ID

### 控制器与功能之间的是如何对应的？

在这个作业中，控制器主要体现在express的路由上，路由负责调用数据库操作的接口，也就是调用Model，各种Model则负责与数据库沟通，操作数据，体现各种业务逻辑。

### 权限管理的UI设计如何解决树的直观性与树难以二维显示的两难境地、而使用HTML table显示又存在不够直观的问题

HTML Table也可以设计得很直观的，可以看我具体的设计，就是占用空间比较大 = =

[TOC]

## 数据库设计文档

### 如何实现数组字段的CRUD？

由于MongoDB不必遵从第一范式，所以增删查改自需要一次调用接口就行了。

### 数组字段vs多字段的性能差别？

数据字段读取快，可以一次性读取全部需要的信息，有效地减少硬盘读写，但是在硬盘空间消耗上较多，冗余度很高，是典型的时空权衡，比较适合MongoDB这种需要减少硬盘读写的文档型数据库。

多字段查询上稍微慢，但是大大减少了空间冗余度。但是在保持数据一致性上有少许困难。但是现代关系型数据库软件非常成熟，因此多字段查询在大型项目里非常常见。

### REST API是怎样设计用于CRUD操作的

RESTFUL API是一种接口设计风格，一般用HTTP幂等动词GET和PUT对应幂等的数据库操作查和改，用HTTP操作POST和DELETE对应数据库操作增加和删除。

而RESTFUL API的路径则对应数据库的某张表或者某个文档(在NoSQL中)。

进一步更加复杂的操作可用子路径和URL query参数来区分，实现参数化查询。

### 各种表的参考完整性是如何保证的

本次实验的参照完整性是借助NodeJS模块mongoose-auto-increment实现的，它有自动增长和模拟外键关联两种功能，同时也保证了参照完整性。

### 如何用SQL的 AO / window function / subquery解决查询问题？

MongoDB在NodeJS里的查询主要是通过异步控制和参数查询来解决查询问题的。

对于多个查询条件，我们可以一步一步查出结果来。

### 如何防止外部导入数据时流水号的冲突

mongoose-auto-increment同时保证了没有流水号冲突，而且也不存在外部导入数据这种功能啊

### 各种数据表是如何备份的

没有数据表备份功能。

### pool\_disagrees这样的字段用静态字段表示和用动态字段表示有何差别？

动态字段消耗时间性能，静态字段消耗空间性能，且不好保证一致性。在MongoDB这种数据库中，我推荐使用静态字段，虽然更加浪费空间，但是能够减少硬盘读写，减小对硬盘的伤害。

### 使用db adapter 还是db ORM？

后者，ORM模型的简单性简化了数据库查询过程。使用ORM查询工具，用户可以访问期望数据，而不必理解数据库的底层结构。我们这只是一个作业而已。

### 关于真正的权限管理是一个有向无环图而不是树

因为现实环境中，父角色如公司老总不一定有子角色如公司CEO的权限，这种权限分离是很常见的，如果是树的话，父角色包裹了子角色，应该有子角色的所有权限才对，两者矛盾。因此是一个有向无环图，而不是退化为树。

### 关于精简掉的那些信息

有些信息如注册时的性别等信息，实际系统中根本用不到，属于数据库的冗余设计，所以我精简掉了。

另外，党费帖老师没有给出对应的模型，所以我也精简掉了。