**Data access layer**

**主要是对非原始数据（数据库或者文本文件等存放数据的形式）的操作层，而不是指原始数据，也就是说，是对数据库的操作，而不是数据，具体为业务逻辑层或表示层提供数据服务**

**1. URL**

url **=** {  
 **"sum" ： "../data/sum/0\_1\_sum.csv"**，  
 **"detail" ： "../data/detail/0\_1\_2\_checkinDetail.csv"**，  
 **"randomdetail" ： "../data/random\_detail/0\_1\_randomcheckinDetail.csv"**，  
 **"seq" ： "../data/seq.csv"**，  
 **"lea" ： "../data/lea.csv"**，  
 **"setting" ： "../interior/settings.ini"**，  
 **"courseInfo" ： "../interior/courseInfo.csv"**，  
 **"teacherInfo" ： "../interior/teacherInfo.csv"**，  
 **"studentInfo" ： "../interior/studentInfo.csv"**，  
}

**2. Read**

**readfile（self， *filename*， *args*=None）**

* **支持以参数传入组成动态路径的写入方式获取目标路径**
* **支持以固定传入的方式获取目标路径**
* **支持各种类型的文件读取错误检测**

**3. Write**

**writefile（self， *data*， *filename*， *args*=None， *type*=None）：**

* **支持以参数传入组成动态路径的写入方式获取目标路径**
* **支持以固定传入的方式获取目标路径**
* **支持w+**
* **支持a+**

**4. Newfile**

**newfile**（self， *filename*， *args***=None**）**：**

* **生成新文件**
* **检测是否已存在**
* **自动Header填充**

**5. Delfile**

* **按需求删除文件**

**6. Delline**

* **按需求删除行**

**7. AlterItem**

* **按需求更改数据项**

**Business Logic Layer**

**主要是针对具体的问题的操作，也可以理解成对数据层的操作，对数据业务逻辑处理，如果说数据层是积木，那逻辑层就是对这些积木的搭建**

**Csvkit**

**https：//docs.python.org/2.7/library/csv.html#module-csv**

**https：//docs.python.org/3.6/library/csv.html#module-csv**

**PRE**

**1. isSucc三种状态**

**针对学生考勤提交的特征信息内容，通过系统内部的考勤识别机制可根据其识别、认证的结果成功与否来响应本次学生提交的特征信息是否有效**

**1）true： 认证成功，即进行3，并修改isSucc状态**

**2）false： 认证失败，不可进行3，仅修改isSucc状态**

**3）空： 未参与**

**2. 出勤有效时间界定机制（starttime， checkinTime）**

**~~1）分类~~**

**~~2）聚类~~**

**3）2分钟内**

**3. 考勤结果算法**

**在学生考勤时，根据late checkinResult和late checkinTime，计算最终的checkinResult**

**根据2的出勤有效阶段判定机制： ^在出勤有效时间范围内 ^^不在**

**Detail.csv中学生的考勤状态： 最新的状态**

**1）出勤： ^keep ^^迟到**

**2）缺勤： ^迟到 ^^keep**

**3）迟到： ^keep ^^早退**

**4）早退： ^迟到 ^^keep**

**5） 空： ^出勤 ^^迟到**

**~~6）请假~~**

* **因此，学生考勤的过程为 1 -> 3**

**0. 辅助**

**0.1 获取模块**

**0.1.1 获取本次考勤次序号 （course\_id）**

**0.1.2 获取teacher\_id （wechat\_id）**

**0.1.3 获取teacher\_id （course\_id）**

**0.1.4 获取student\_id （wechat\_Id）**

**0.1.5 获取学生所有课程号 （wechat\_id）**

**0.1.6 获取教师所有课程号 （wechat\_id）**

**0.1.7 获取班级名列表（course\_id）**

**0.1.8 获取timewindow （url）**

**0.1.9 获取某课程的所有学生学号（course\_id）**

**0.1.10 获取当前时间 （）**

**0.2 初始化detail.csv （course\_id）**

**1）0.1.3 0.1.1+1 初始化文件（DAL4，detail）**

**获取当前课程的所有学号，checkintype为auto，其余的为空**

**0.3 初始化randomdetail.csv（course\_id， random\_list）**

**1）若没有此teacherid\_random.csv，则根据0.1.2来创建新文件，添加header信息**

**2）清空除header之外的信息（仅用一次，结束后需更新至detail.csv）**

**3）根据random\_list为stuid赋值，将checkintype置auto，其余为空**

**0.4 空太多（course\_id）**

**1）读取detail.csv**

**2）若checkinResult的空状态的数量超过总体的百分之80，则此次考勤无效**

**0.6 计算考勤结果（course\_id）**

**针对该course\_id课程：**

**1）检验：4.9、至少存在一次考勤记录、是否有请假记录**

**2）遍历teacherid\_lea.csv的内容，若存在与course\_id相关的请假记录，若教师给予请假，并将请假状态更新至相应的detail.csv文件，若不给予请假则不更新**

**3）若教师忽略批假则将对应学生的考勤状态置在sum.csv中置为“空”，直到教师处理该学生的假条**

**4）遍历course\_id下的所有detail.csv，并将考勤结果写入sum.csv**

**0.7 合并考勤结果 （course\_id）**

**1）获取源url与目标url格式[0.1.3 course\_id 0.1.1]**

**2） 若random.csv文件中包含有效数据，则根据randomdetail.csv更新detail文件的内容，若checkinresult不同，则以random.csv为主，即新的checkinresult覆盖detail.csv**

**0.8 生成random列表（nums）**

**若有学生请假成功，则不会出现在此random列表中， 先判断请假名单是否有此学生， 若有，则删除。 此学生不用考勤，但占用nums一个坑。**

**0.9 格式化统计结果（course\_id）**

**打印出此课程对应每个人的考勤状况，每节课出勤率，平均出勤率**

**0.10 更新sum.csv**

**提取detail.csv的checkinresult，结果更新至sum.csv （方便查询）**

**0.11 更新seq.csv（course\_id）**

**首先获取下一次的seq.csv**

**1. 检验**

**1.1 课程号检验（wechat\_id， course\_id）**

**1.2 抽点人数检验 （num）**

**1.3 考勤状态检验（type）**

**1.4 考勤次序号检验（seq） 是否存在**

**1.5 课程号学号检验（stu\_id）**

**1.6 教师微信号检验 （wechat\_id）**

**1.7 学生微信号检验 （wechat\_id）**

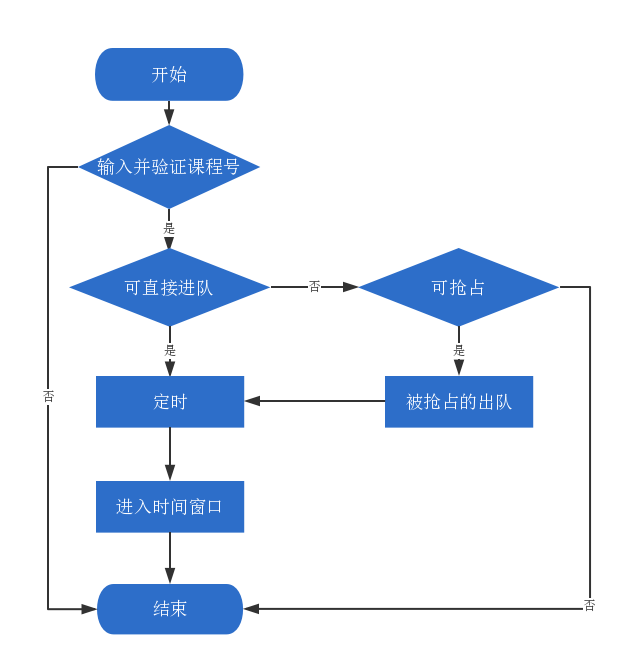
**2. 教师**

**2.0 查看我的course\_id （wechat\_id）**

**1）检验：1.6**

**2）0.1.6**

**2.1 自助考勤（wechat\_id，course\_id，~~timer\_list~~） 教师入队是互斥的--threading.RLock**

****

**1）检验：1.1、1.6**

**2）互斥的进入4.5**

**3）合并被抢占的抽点考勤详细表。**

**3）维护detail.csv**

**4）维护seq.csv**

**2.2 抽点（wechat\_id，  ~~timer\_list~~） （在时间窗口自动检测courseid且支持多次开启抽点）**

**1）检验： 1.6、4.7、当前时间小于4.8-int（5\*60）**

**2）若教师再次开启抽点（获取时间窗口的抽点次数标志，若重复开启抽点考勤则合并（merge 0.7（0.1.3））。**

**3）输入抽点人数1.2 初始化0.3（4.10， 0.8（course\_id））**

**4）维护seq.csv**

**5）初始化时间窗口的抽点考勤初始时间**

**2.3 手工（wechat\_id，course\_id）**

**1）检验：1.1、1.6、非4.7**

**2）维护detail.csv，默认全部出勤**

**3）教师可输入学号1.5和状态1.3，**

**4）更新detail 自动获取seq 调用0.1 维护seq.csv**

**2.4 修改（wechat\_id，course\_id，seq\_id）**

**1）检验：1.1、1.4、1.6、非4.7**

**2）输入学号1.5+状态1.3，**

**3）更新detail.csv**

**~~3）更新sum.csv~~**

**2.5 汇总 （wechat\_id， course\_id）**

**1）检验：1.6、非4.7**

**2）0.6 ~~想要汇总某个课程，必须先把所有课程假条处理完毕~~**

**3）0.9**

**2.6 查看最近 （wechat\_id） 传入后，显示**

**1）检验：1.6**

**2）直接seq中的信息，获取相关url得到detail**

**3）统计：出勤率，缺勤率等**

**3. 学生**

**3.0 查看我的course\_id （wechat\_id）**

**1）检验：1.7**

**2）0.1.5**

**3.1 考勤 （wechat\_id）**

**1）检验：1.6、1.7、1.8 考勤成功后不可重复考勤（自助，抽点）**

**2）上传特征信息**

**3）参考开头的pre**

**3.2 请假 （wechat\_id） [注]可以帮忙请假，因此此处的wechat\_id不具有完全的充分性**

**1）检验：1.6、1.7、1.8 请假成功后可重复请假 一分钟以内若提交假条三次，则禁用此功能5分钟**

**2）上假条lea**

**3）6.1**

**3.3 查看正在进行的考勤 （wechat\_id， timer\_list）**

**1）检验： 1.7、1.8**

**2）根据timer\_list中对应的course\_Id， detail.csv、randomdetail.csv（优先）相关内容**

**3.4 查看历史考勤 （wechat\_id，course\_id）**

**1）检验：1.7**

**2）根据course\_Id， 查询sum.csv相关内容**

**4. Timer  
 TCB=[wechat\_id ，course\_id， sectime， endtime，class\_list[]]**

**Timer\_list=[] 存放TCB 顶层的全局变量**

**4.1 获取sectime （url）上课区间包括开始和结束**

**1）获取当前时间，获取sectime列表**

**2）若当前时间，小于sectime列表第一个或大于其最后一个，返回sec1**

**3）根据sec1，2，3匹配，若起始sec<=若当前时间<结束sec， 返回对应secn. 否则：若当时间小于最后一节课的起始sec，若并且当前时间大于等于secn的结束时间并且小于secn+1的开始时间，返回secn+1 否则，返回sec\_last**

**4.3 初始化tcb（wechat\_id， course\_id）**

**[wechat\_id ， course\_id， 4.1， 0.1.6]**

**4.4 进队检测与处理（~~wechat\_id ，~~ course\_id）**

**1）若timer\_list空，返回true**

**2）根据course\_id获取班级名列表（4.3）， 获取new\_sectime（4.1）**

**3）获取的班级名列表和timer\_list中的每个TCB[]中的class\_list[]依次进行查重操作**

**Set（list1） & Set（list2），若没有重复，返回true. 否则，获取出现重复班级名的sectime与new\_sectime对比，若相同，返回false并提示失败原因. 若不同，返回true并且获取出现重复班级名的course\_id， TCB直接出队（4.6） ：并且由于将要出队的tcb是被抢占而异常出队，因此需要使用merge（course\_id）来合并此课程号的抽点考勤详细表与详细表的内容后，才可以出队。**

**4.5 进队（wechat\_id， course\_id）**

**1）进队检测与处理（4.4），**

**2）4.3**

**3）TCB进队，threading.Timer（0.1.7， 4.6， （"msg1"，"msg2"））**

**[注] threading.Timer（1， self.depart\_queque， （*wechat\_id*， ））**

**4.6 出队（wechat\_id，course\_id）**

**1）若timerlist中由相同的班级名0.1.7，若4.7真，继续**

**2）考虑到教师在被抢占时出队，为防止Timer重复执行出队函数，若4.12为false，**

**3）TCB出队**

**4.7 根据wechat检测是否在队列（wechat）**

**1）timerlist存在**

**2）true or false**

**4.8 获取endtime（wechat\_id）**

**4.9 根据course\_id检测是否在队列（course\_id）**

**if not a：**

**print（"List is empty"）**

**4.10根据wechat获取教师课程号**

**4.11TMER计算考勤结果（wechat\_id， course\_id，~~timer\_list）~~**

**以detail.csv为主线，按学号依次计算**

**1）检验：4.7（此处特殊，存在才能继续）**

**2）若0.4，删除该detail.csv，~~初始化random.csv 0.3，~~停止。**

**3）0.7 0.10**

**4.12 判断是否可正常出队（courseid）**

**获取将要出队的courseid对应的班级名列表，再获取当前时间窗口中的tcb对应班级名列表，依次判断两个班级名列表中的数据是否重复。若出现重复， 返回false。否则，返回true。**

**5. 管理员**

**Import\_file + format\_check**

**6. 外部系统**

**6.1 请假 （student\_id， course\_id， seq\_id，lea）**

**1）假条识别系统（lea）**

**2）系统成功识别后，将【student\_id， course\_id， seq\_id】存入teacherid\_lea.csv**

**User Interface Layer**

**Function provided for users**