**考勤计算二次开发**

**帮助文档**

目录

[一. 文档概述 4](#_Toc456013163)

[二. 流程一览 5](#_Toc456013164)

[1． 考勤计算入口 6](#_Toc456013165)

[2． 考勤计算准备数据 6](#_Toc456013166)

[2.1考勤计算-计算器封装 6](#_Toc456013167)

[2.2.1 考勤计算-计算器（calculator） 6](#_Toc456013168)

[2.2 全局信息集globeMap封装 7](#_Toc456013169)

[2.2.1 与人员无关的基础信息集 7](#_Toc456013170)

[2.2.2与人员有关的动态信息集 7](#_Toc456013171)

[3． 内置考勤项目计算 8](#_Toc456013172)

[3.1 取卡 8](#_Toc456013173)

[3.2 内置考勤项目计算（打卡） 9](#_Toc456013174)

[3.2.1 异常 9](#_Toc456013175)

[3.2.2 请假 10](#_Toc456013176)

[3.2.3 出差 10](#_Toc456013177)

[3.2.4 加班 11](#_Toc456013178)

[3.2.5 其他 11](#_Toc456013179)

[3.3 内置考勤项目计算（不打卡） 12](#_Toc456013180)

[3.3.1 请假 12](#_Toc456013181)

[3.3.2 出差 13](#_Toc456013182)

[3.3.3 加班 13](#_Toc456013183)

[3.3.4 其他 13](#_Toc456013184)

[4． 自定义考勤项目计算 14](#_Toc456013185)

[4.1 考勤计算公式 14](#_Toc456013186)

[4.2 自定义函数 14](#_Toc456013187)

[三. 示例演示 15](#_Toc456013188)

[3.1 前端页面配置 15](#_Toc456013189)

[3.2 后端重构 16](#_Toc456013190)

[四. 总结 17](#_Toc456013191)

[附录1.考勤计算常量表 18](#_Toc456013192)

[附录2.考勤计算信息集速查 27](#_Toc456013193)

[附录2.1 Map Param 信息集 结构表 27](#_Toc456013194)

[附录2.2 calculateProParamsVo 对象 结构表 28](#_Toc456013195)

[附录2.3 globeMap 信息集 结构表 30](#_Toc456013196)

# 文档概述

考勤计算是考勤业务中的重要一环，随着代码的不停迭代，目前的考勤计算功能已经能够高效准确地完成既定的任务。但是，考虑到业务的多样性和延展性，出厂的考勤计算功能可能不能100%地满足每个公司的业务需求，为了让广大客户能够更全面的使用和更个性化地定制自己的业务，在最新版本的系统中，考勤计算已经能具有一定继承性和扩展性，

可以进行深度的二次开发。

所谓的二次开发，就是基于目前考勤计算框架，在现有的软件上进行定制修改，功能的扩展，然后达到自己想要的功能。而考勤计算的二次开发，主要包括以下两方面的内容：

1. 考勤内置项目计算的重写开发；2、考勤自定义项目自定义函数的扩展开发；

所以接下来文档将会全面地介绍考勤计算的框架，并会举例说明二次开发的流程以及各大注意事项。

在阅读之前我们希望您已经具有一定的考勤计算业务基础，也有使用JAVA语言的实战经验，这样会更加快速地吸收文档内包含的知识。

# 流程一览



# 考勤计算入口

目前考勤计算有四个入口，分别是：（1）考勤计算看板；（2）考勤计算页面-计算选中行；（3）考勤计算按页面-全部计算；（4）考勤计算后台事务。

这四个考勤计算入口后台走的都是同一套逻辑，所有二次开发无须针对四个入口分别定制，进行一次的二次开发即对所有的入口同时生效。

# 考勤计算准备数据

## **2.1**考勤计算-计算器封装

会根据当前系统中的所有考勤制度上的设置，将各个业务的计算器封装在一起（有的业务分为打卡考勤/非打卡考勤）；考勤计算过程中，会根据当前计算人员对应的考勤制度以及打卡类型（打卡考勤/非打卡考勤），自动选择不同的计算器集合依次调用集合中的计算器。

### 2.2.1 考勤计算-计算器（calculator）

考勤计算的计算器是二次开发的主要内容，它包含了考勤计算全部内置项目的计算，一个业务可能有一到两个计算器，每个计算器对应了一个类，每个类中计算了多个内置考勤项目（可以在附录中查看）在考勤制制度中各块业务的选项卡中可以配置，依次为：

1. 异常：只有一个计算器，只会在打卡考勤中生效；
2. 请假：分为打卡考勤/不打卡考勤两个计算器；
3. 出差：分为打卡考勤/不打卡考勤两个计算器；
4. 加班：分为打卡考勤/不打卡考勤两个计算器；
5. 取卡：只有一个计算器，只会在打卡考勤中生效；
6. 其他：分为打卡考勤/不打卡考勤两个计算器；

配置完毕后，实际考勤计算的时候，会根据人员是否打卡/不打卡考勤，组装不同的计算器。

配置的内容为对应业务类的全路径，二次开发的时候，可以对这些类进行继承，在子类上进行二次开发，或者定义一个全新类，然后将完成之后的类全路径，复制到对应的配置项中。

在二次开发的计算器中，每个计算器必须返回对应业务计算的内置考勤项目，否则计算结果将有缺失，附录中可以查看每个业务对应的计算器必须返回的结果集内容。

**注意：严禁修改出厂计算器对应的后台类。**

## 2.2 全局信息集globeMap封装

### **2.2.1 与人员无关的基础信息集**

在考勤计算数据准备阶段，会准备这一部分的数据，并把结果放进globeMap中，在整个考勤计算过程中，这部分信息集不会做更变也不会做更新。属于静态信息集。所以在考勤计算过程中，请勿更改一下相关的设置，以免造成考勤计算结果和设置不一致的情况。

信息集包括：

1.批量查询取卡规则单个信息的集合；

2.批量查询所有班次信息集合；

3.所有假期制度；

4.所有考勤制度；

5.是否参与调休计算参数-公共参数；

6.是否参与加班管控计算参数-公共参数；

7.是否参与加班管控计算参数-加班管控 ；

8.全局默认的工作日历；

9.查询工作日历明细数据；

10.假期制度主表信息；

11.查询以下考勤基础资料：

加班类型、出差类型、假期类型、班次类型、加班原因、加班补偿方式。

注：在附录中可以查询信息集的详情。

### 2.2.2与人员有关的动态信息集

进入考勤计算主流程后，在进行具体业务计算前，会进行与人员有关的信息集查询，这部分信息集称为动态信息集，查询完这部分数据后，会结合静态信息集 ，对考勤计算人员进行过滤，拆分打卡/不打卡考勤人员，记录不参与计算人员的错误日志等。

信息集包括：

1、批量查询考勤档案信息集合；

2、查询一批次人员的 人员职位,岗位,组织,HR组织-信息集合；

3、查询一批次人员的 任职历史,查询某人考勤日期那天的组织；

4、查询一段时间,一批人次的原始打卡记录信息；

5、查询人员和考勤日期对应的考勤明细信息；

6、查询人员和考勤日期对应的排班分录信息；

7、与人，考勤计算开始结束时间有关的所有单据信息：

包括出差单、加班单、请假单、请假单明细、销假单、补签卡；

8、在最后，会根据已有的信息集信息，做一次过滤和分类，排除掉不能考勤计算的人员，并记录日志，将打卡和非打卡考勤人员分类。

注：在附录中可以查询信息集的详情。

# 内置考勤项目计算

## 3.1 取卡

对应的计算器为：com.kingdee.shr.ats.calculate.implement.FetchCardPro

此计算器必须对以下结果集进行计算更新：

|  |  |
| --- | --- |
| **结果集变量名** | **变量含义** |
| oneOnTime | 第一段上班时间 |
| oneOffTime | 第一段下班时间 |
| twoOnTime | 第二段上班时间 |
| twoOffTime | 第二段下班时间 |
| threeOnTime | 第三段上班时间 |
| threeOffTime | 第三段下班时间 |
| oneOnWorkTimeEarliest | 一段段头最早时间 |
| oneOnWorkTimeLatest | 一段段头最晚时间 |
| oneOffWorkTimeEarliest | 一段段尾最早时间 |
| oneOffWorkTimeLatest | 一段段尾最晚时间 |
| twoOnWorkTimeEarliest | 二段段头最早时间 |
| twoOnWorkTimeLatest | 二段段头最晚时间 |
| twoOffWorkTimeEarliest | 二段段尾最早时间 |
| twoOffWorkTimeLatest | 二段段尾最晚时间 |
| threeOnWorkTimeEarliest | 三段段头最早时间 |
| threeOnWorkTimeLatest | 三段段头最晚时间 |
| threeOffWorkTimeEarliest | 三段段尾最早时间 |
| threeOffWorkTimeLatest | 三段段尾最晚时间 |

## 3.2 内置考勤项目计算（打卡）

### 3.2.1 异常

对应的计算器为：com.kingdee.shr.ats.calculate.implement.PunchardExceptionPro

此计算器必须对以下结果集进行计算更新：

|  |  |
| --- | --- |
| **结果集变量名** | **变量含义** |
| oneLateValue | 第一段迟到分钟 |
| oneLeaveValue | 第一段早退分钟 |
| twoLateValue | 第二段迟到分钟 |
| twoLeaveValue | 第二段早退分钟 |
| threeLateValue | 第三段迟到分钟 |
| threeLeaveValue | 第三段早退分钟 |
| oneAbsentValue | 第一段旷工分钟 |
| twoAbsentValue | 第二段旷工分钟 |
| threeAbsentValue | 第三段旷工分钟 |
| lateTimes | 迟到次数 |
| lateMinuts | 迟到分钟 |
| leaveTimes | 早退次数 |
| leaveMinuts | 早退分钟 |
| absentTimes | 旷工次数 |
| absentHours | 旷工小时 |
| lackCardTimes | 缺卡次数 |

### 3.2.2 请假

对应的计算器为：com.kingdee.shr.ats.calculate.implement.PunchardLeaveBillPro

此计算器必须对以下结果集进行计算更新：

|  |  |
| --- | --- |
| **结果集变量名** | **变量含义** |
| leaveAtsTimes | 请假次数 |
| leaveAtsHours | 请假时长(小时) |
| leaveAtsDays | 请假时长(天) |
| sickLeaveTimes | 病假次数 |
| sickLeaveDays | 病假时长（天） |
| sickLeaveHours | 病假时长（小时） |
| yearLeaveTimes | 年假次数 |
| yearLeaveDays | 年假时长（天） |
| yearLeaveHours | 年假时长（小时） |
| matterLeaveTimes | 事假次数 |
| matterLeaveDays | 事假时长（天） |
| matterLeaveHours | 事假时长（小时） |
| tuneLeaveDays | 调休假时长(天) |
| tuneLeaveHours | 调休假时长(小时) |
| marryLeaveDays | 婚假时长(天) |
| birthDays | 产假时长(天) |
| accompanyBirthDays | 陪产假时长(天) |
| birthControlDays | 节育假时长(天) |
| injuryDays | 工伤假时长(天) |
| visitDays | 探亲假时长(天) |
| funeralDays | 丧假时长(天) |
| otherLeaveDays | 其他假期时长(天) |
| otherLeaveHours | 其他假期时长(小时) |
| checkLeaveHours | 产检假时长(天) |

### 3.2.3 出差

对应的计算器为：com.kingdee.shr.ats.calculate.implement.PunchardTripBillPro

此计算器必须对以下结果集进行计算更新：

|  |  |
| --- | --- |
| **结果集变量名** | **变量含义** |
| tripTimes | 出差次数 |
| tripDays | 出差时长（天） |
| tripHours | 出差时长（小时） |

### 3.2.4 加班

对应的计算器为：com.kingdee.shr.ats.calculate.implement.PunchardOverTimeBillPro

此计算器必须对以下结果集进行计算更新：

|  |  |
| --- | --- |
| **结果集变量名** | **变量含义** |
| tuneOT1Hours | 调休OT1小时数 |
| tuneOT2Hours | 调休OT2小时数 |
| tuneOT3Hours | 调休OT3小时 |
| overTimeOT1Hours | 计加班费OT1小时数 |
| overTimeOT2Hours | 计加班费OT2小时数 |
| overTimeOT3Hours | 计加班费OT3小时数 |

### 3.2.5 其他

对应的计算器为：com.kingdee.shr.ats.calculate.implement.PunchardOtherPro

此计算器必须对以下结果集进行计算更新：

|  |  |
| --- | --- |
| **结果集变量名** | **变量含义** |
| dayHour | 应出勤时数-每天工作小时数 |
| dayHourFact | 实际出勤时数 |
| attendDays | 应出勤天数 |
| attendDaysFact | 实际出勤天数 |
| fillTimes | 补卡次数 |
| signTimes | 签卡次数 |

## 3.3 内置考勤项目计算（不打卡）

### 3.3.1 请假

对应的计算器为：com.kingdee.shr.ats.calculate.implement.NoPunchardLeaveBillPro

此计算器必须对以下结果集进行计算更新：（同打卡）

|  |  |
| --- | --- |
| **结果集变量名** | **变量含义** |
| leaveAtsTimes | 请假次数 |
| leaveAtsHours | 请假时长(小时) |
| leaveAtsDays | 请假时长(天) |
| sickLeaveTimes | 病假次数 |
| sickLeaveDays | 病假时长（天） |
| sickLeaveHours | 病假时长（小时） |
| yearLeaveTimes | 年假次数 |
| yearLeaveDays | 年假时长（天） |
| yearLeaveHours | 年假时长（小时） |
| matterLeaveTimes | 事假次数 |
| matterLeaveDays | 事假时长（天） |
| matterLeaveHours | 事假时长（小时） |
| tuneLeaveDays | 调休假时长(天) |
| tuneLeaveHours | 调休假时长(小时) |
| marryLeaveDays | 婚假时长(天) |
| birthDays | 产假时长(天) |
| accompanyBirthDays | 陪产假时长(天) |
| birthControlDays | 节育假时长(天) |
| injuryDays | 工伤假时长(天) |
| visitDays | 探亲假时长(天) |
| funeralDays | 丧假时长(天) |
| otherLeaveDays | 其他假期时长(天) |
| otherLeaveHours | 其他假期时长(小时) |
| checkLeaveHours | 产检假时长(天) |

### 3.3.2 出差

对应的计算器为：

com.kingdee.shr.ats.calculate.implement.NoPunchardTripBillPro

此计算器必须对以下结果集进行计算更新：（同打卡）

|  |  |
| --- | --- |
| **结果集变量名** | **变量含义** |
| tripTimes | 出差次数 |
| tripDays | 出差时长（天） |
| tripHours | 出差时长（小时） |

### 3.3.3 加班

对应的计算器为：com.kingdee.shr.ats.calculate.implement.NoPunchardOverTimeBillPro

此计算器必须对以下结果集进行计算更新：（同打卡）

|  |  |
| --- | --- |
| **结果集变量名** | **变量含义** |
| tuneOT1Hours | 调休OT1小时数 |
| tuneOT2Hours | 调休OT2小时数 |
| tuneOT3Hours | 调休OT3小时 |
| overTimeOT1Hours | 计加班费OT1小时数 |
| overTimeOT2Hours | 计加班费OT2小时数 |
| overTimeOT3Hours | 计加班费OT3小时数 |

### 3.3.4 其他

对应的计算器为：com.kingdee.shr.ats.calculate.implement.NoPunchardOtherPro

此计算器必须对以下结果集进行计算更新：

|  |  |
| --- | --- |
| **结果集变量名** | **变量含义** |
| dayHour | 应出勤时数-每天工作小时数 |
| dayHourFact | 实际出勤时数 |
| attendDays | 应出勤天数 |
| attendDaysFact | 实际出勤天数 |

**注：请勿对所有内置考勤项目的出厂计算器进行任何形式的修改。**

# 自定义考勤项目计算

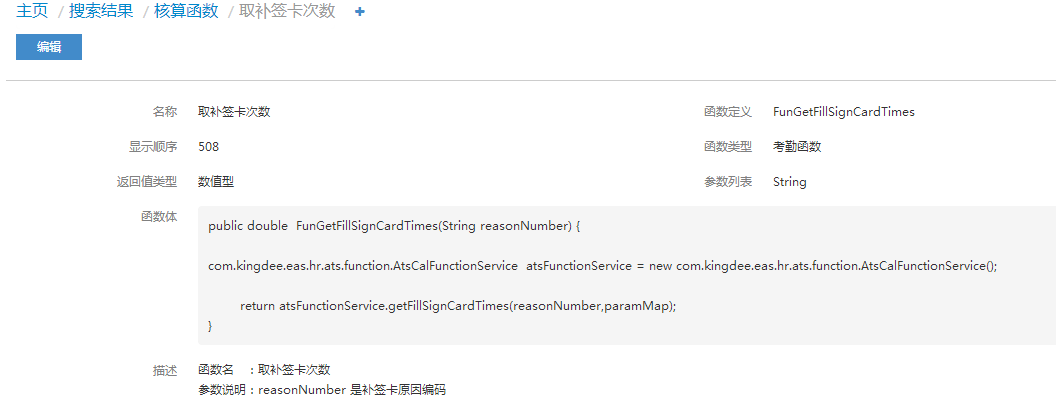
自定义的考勤项目计算是基于公式平台的，通过定制自定义公式，调用自定义函数实现考勤结果的二次运算。

## 4.1 考勤计算公式

在【考勤制度】- 点击某个考勤制度-【计算公式】页面可以配置自定义公式，自定义公式是基于自定义函数的，通过一个或多个函数的组合运算，得到自定义公式。

## 4.2 自定义函数

自定义函数在用administrator登录后，在【核算函数】页面配置，如以下画面所示：



目前【核算函数】内置了若干个考勤相关的函数，对于内置函数不能满足需求的，可以进行自定义函数的二次开发。

一个函数对应了后台类的一个方法， 若要自定义函数的开发，首先必须进行后台方法的实现，然后再在【核算函数】页面进行函数的注册配置。

函数体的书写格式参考上图传入参数必须和实现方法一一对应。

**注：禁止修改内置的自定义函数。**

# 示例演示

接下来将展示考勤计算那种【异常】业务相关的二次开发。

# 3.1 前端页面配置

1、首先打开【考勤制度】，选中要进行二次开发的考勤制度并打开页面。

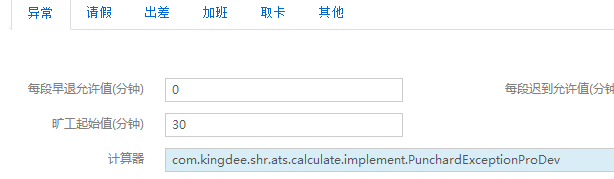


2、可以看到，默认的【异常】业务的计算器是：com.kingdee.shr.ats.calculate.implement.PunchardExceptionPro

3、我们假设二次开发后的计算器名称为：PunchardExceptionProDev

4、点击【编辑】按钮，将以下内容覆盖原本的计算器：

com.kingdee.shr.ats.calculate.implement.PunchardExceptionProDev



# 3.2 后端重构

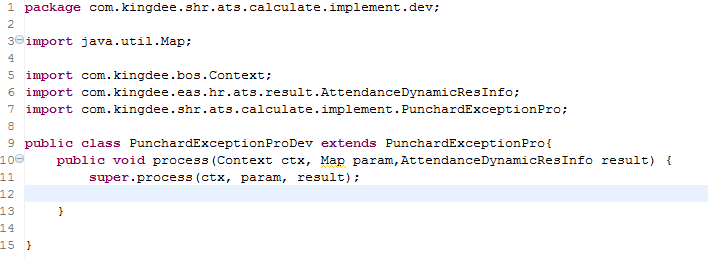
1、新建一个包，假设为：com.kingdee.shr.ats.calculate.implement.dev

**注：包路径可根据自身需求定义，但不可和出厂计算器包一致，以免造成代码紊乱。**

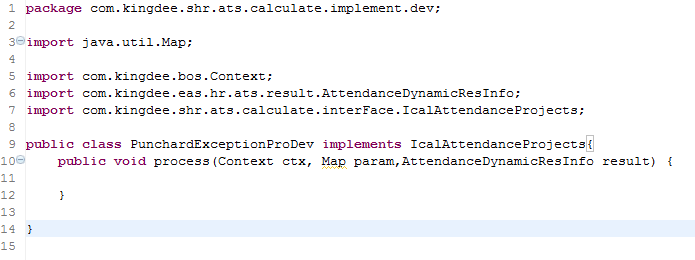
（出厂包路径com.kingdee.shr.ats.calculate.implement）

1. 在此路径下新建类，类名设定为刚才我们在前端页面配置的计算器名字PunchardExceptionProDev

2.1我们可以基于原本逻辑进行继承，如下图所示：

 若要对少部分的逻辑进行二次开发，建议使用继承模式，可沿用父类大部分的实现，在进行子类的开发之前，可先调用一次父类逻辑，对父类结果集进行二次更新。

2.2我们也可以进行全部逻辑重写，必须实现接口IcalAttendanceProjects，如下图所示



1. 接下来可以在此类中进行逻辑重载/重构。在以上两种中，无论是继承或是实现模式，都定义了方法：

public void process(Context ctx, Map param,AttendanceDynamicResInfo result);

其中 参数ctx 表示 上下文，这里有当前登录用户信息和数据库的连接信息，若要在二次开发计算器中进行数据库连接，可使用这个参数，但实际为了考虑性能，我们不建议在二次开发过程中进行新的数据库连接；

Param 参数 内包含了考勤计算过程中要用到的信息集，文中我们称为globeMap；

Result 参数是所有内置考勤项目的结果集，考勤计算的过程中，每个计算器对其中的一个或者多个参数进行了更新，最后会统一保存。

**注：Param和resulst 的具体结构和内容可参照附录。**

1. 最后必须更新【异常】业务对应的result 结果集，通过查阅上文，我们知道异常业务的参数包括：AbsentHours，AbsentTimes，LateMinuts，LateTimes，LeaveMinuts，LeaveTimes，OneAbsentValue，OneLateValue，OneLeaveValue，TwoAbsentValue，TwoLateValue，TwoLeaveValue，ThreeAbsentValue，ThreeLateValue，ThreeLeaveValue,lackCardTtimes；

在计算过程中我们可能会使用到结果集result的部分参数，我们为每个参数定义了取参数的方法getXXX和设参数的方法setXXX，为了代码规范和安全：

如果我们取AbsentHours这个参数，我们使用result.getAbsentHours()这个方法，而不是result. absentHours；

如果我们要设AbsentHours 这个参数，我们使用result.set AbsentHours(XXX)而不是result. absentHours = XXX。

1. 【异常】业务的重构就完成。

# 总结

本文档从框架出发，介绍了考勤计算二次开发的整个流程，并对流程中的每一个步骤进行了详细的讲解，最后对讲解内容进行了实例化，帮助理解消化文中内容。

此后，随着考勤计算功能不断强大，二次开发文档也会不断完善。充实其中内容和修改其中的错误。

请开发人员进行二次开发前，务必仔细研读此文档。

# 附录1.考勤计算常量表

对应位置为：**com.kingdee.eas.hr.ats.common.AttendanceConstants**

**请使用AttendanceConstants.XXX的形式使用考勤计算的常量**

/\*\*

\* 下划线分隔符<br>

\*/

public static final String UNDERLINE = "\_";

/\*\*

\* 批量查询所有请假单据信息集合,人员\_日期作为key Map<String, List<AtsLeaveBillEntryInfo>>

\*/

public final static String ATSLEAVEBILLINFOSMAP = "T\_HR\_ATS\_LeaveBillEntry" ;

/\*\*

\* 批量查询所有出差单据信息集合,人员\_日期 作为key Map<String, List<AtsTripBillEntryInfo>>

\*/

public final static String ATSTRIPBILLINFOSMAP = "T\_HR\_ATS\_TripBillEntry" ;

/\*\*

\* 批量查询所有加班单据信息集合,人员\_日期 作为key Map<String, List<AtsOverTimeBillEntryInfo>>

\*/

public final static String ATSOVERTIMEBILLINFOSMAP = "T\_HR\_ATS\_OverTimeBillEntry" ;

/\*\*

\* 批量查询所有补签卡单据信息集合,人员\_日期作为key Map<String, List<FillSignCardInfo>>

\*/

public final static String FILLSIGNCARDINFOSMAP = "fillSignCardInfosMap" ;

/\*\*

\* 装载请假单据明细信息集合,人员\_日期 作为key Map<String,List<AtsLeaveBillDetailInfo>>

\*/

public final static String ATSLEAVEBILLDETAILINFOSMAP = "atsLeaveBillDetailInfosMap" ;

/\*\*

\* 批量查询取卡规则单个信息的集合,取卡规则的ID作为key Map<String, CardRuleInfo>

\*/

public final static String CARDRULESINGLEINFOMAP = "T\_HR\_ATS\_CardRule" ;

/\*\*

\* 批量查询所有班次信息集合,班次主表ID作为key Map<String, AtsShiftInfo>

\*/

public final static String ATSSHIFTINFOMAP = "T\_HR\_ATS\_Shift" ;

/\*\*

\* 查询所有假期制度的ID(BOSUUID)作为key Map<String, HolidayPolicyInfo>

\*/

public final static String HOLIDAYPOLICYMAP = "T\_HR\_ATS\_HolidayPolicy" ;

/\*\*

\* 查询所有考勤制度的ID(BOSUUID)作为key Map<String, AttencePolicyInfo>

\*/

public final static String ATTENDANCEPOLICYINFOMAP = "T\_HR\_ATS\_AttencePolicy" ;

/\*\*

\* 查询所有工作日历集合的ID(BOSUUID)作为key 工作日历为key

\* Map<String, WorkCalendarInfo>

\*/

//public final static String WORKCALENDARCOL = "T\_HR\_ATS\_WorkCalendar" ;

public final static String WORKCALENDARINFOMAP = "T\_HR\_ATS\_WorkCalendar" ;

/\*\*\*

\* 全局默认的工作日历 WorkCalendarInfo

\*/

public final static String DEFAULTWORKCALENDARINFO = "defaultWorkCalInfo" ;

/\*\*

\* 查询工作日历明细数据,按照开始日期和结束日期查询 工作日历\_日期作为key Map<String, WorkCalendarItemInfo>

\*/

public final static String WORKITEMMAP = "workItemMap" ;

/\*\*

\* 批量查询考勤档案信息集合,人员\_日期 作为key Map<String, AttendanceFileInfo>

\*/

public final static String ATTENDANCEFILEINFOMAP = "T\_HR\_ATS\_AttendanceFile" ;

/\*\*

\* 查询一批次人员的 人员职位,岗位,组织,HR组织-信息集合,人员ID作为key,考勤计算最后查询-HR组织-等字段使用

\*

\* Map<String,PersonPositionInfo>

\*/

public final static String PPINFOMAP = "ppInfoMap" ;

/\*\*

\* 查询一批次人员的 任职历史,查询某人考勤日期那天的组织 人员ID作为key Map<String,List<EmpPosOrgRelationInfo>>

\*/

public final static String EMPPOSINFOMAP = "empPosInfoMap" ;

/\*\*

\* 查询一批次人员的 变更历史 人员ID作为key Map<String,List<PersonHisInfo>> PersonHisInfoMap

\*/

public final static String PERSONHISINFOMAP = "PersonHisInfoMap" ;

/\*\*

\* 查询一段时间,一批人次的原始打卡记录信息,加班取卡使用 人员ID\_考勤日期 作为key Map<String, List<PunchCardRecordInfo>>

\*/

public final static String PUNCHCARDINFOMAP = "punchCardInfoMap" ;

/\*\*

\* 查询人员和考勤日期对应的考勤明细信息 人员ID\_考勤日期 作为key Map<String, List<AttendanceDynamicResInfo> >

\*/

public final static String ATTENDANCERESULTINFOSMAP = "attendanceResultInfosMap" ;

/\*\*

\* 装载人员和考勤日期对应的排班分录信息,人员\_考勤日期 做为key Map<String, List<ScheduleShiftInfo>>

\*/

public final static String SCHEDULESHIFTITEMMAP = "T\_HR\_ATS\_ScheduleShift" ;

/\*\*\*

\* 开始日期

\*/

public final static String BEGINDATE = "beginDate" ;

/\*\*\*

\* 结束日期

\*/

public final static String ENDDATE = "endDate" ;

/\*\*\*

\* 人员id

\*/

public final static String PERSONID = "personId" ;

/\*\*

\* 当前考勤日期

\*/

public final static String ATTENCEDATE = "attenceDate" ;

/\*\*

\* 手动创建的班前加班单的集合

\*/

public static final String USERCREATE\_PREOTBILLLIST="userCreate\_PreOTBillList";

/\*\*

\* 手动创建的班后加班单的集合

\*/

public static final String USERCREATE\_AFTOTBILLLIST="userCreate\_AftOTBillList";

/\*\*

\* 固定加班生成的加班单集合

\*/

public static final String BACKJOBCREATE\_OTBILLLIST="backJobCreate\_OTBillList";

/\*\*

\* 自动生成班前加班生成的加班单集合

\*/

public static final String BACKJOBCREATE\_PREOTBILLLIST="backjobCreate\_PreOTBillList";

/\*\*

\* 自动生成的班后加班生成的加班单结婚

\*/

public static final String BACKJOBCREATE\_AFTOTBILLLIST="backjobCreate\_AftOTBillList";

/\*\*\*

\* globalMap

\*/

public static final String GLOBALMAP="globalMap";

/\*\*\*

\* 班次的key AtsShiftInfo

\*/

public static final String ATSSHIFTINFO = "atsShiftInfo" ;

/\*\*\*

\* 标准工时的key BigDecimal

\*/

public static final String STANDARHOUR = "standardHour" ;

/\*\*\*

\* 档案 AttendanceFileInfo

\*/

public static final String ATTENCEFILE = "attenceFile" ;

/\*\*\*

\* 考勤制度 AttencePolicyInfo

\*/

public static final String ATTNECEPOLICYINFO = "AttencePolicyInfo";

/\*\*

\* 查询所有加班类型的ID(BOSUUID)作为key Map<String,OverTimeTypeInfo>

\*/

public static final String OVERTIMETYPE = "T\_HR\_ATS\_OverTimeType";

/\*\*

\* 加班补偿方式

\*/

public static final String OVERTIMECOMPENTS = "T\_HR\_ATS\_OverTimeCompens";

/\*\*

\* 个人信息 PersonPositionInfo

\*/

public static final String PERSONPOSITIONINFO = "personPositionInfo";

/\*\*

\* 个人信息 PersonPositionInfo List<EmpPosOrgRelationInfo>

\*/

public static final String EMPPOSORGLIST = "empPosOrgRelationInfo";

/\*\*

\* 个人历史信息 PersonPositionInfo List<PersonHisInfo>

\*/

public static final String PERSONHISINFO = "PersonHisInfo";

/\*\*

\* 个人组织和岗位信息 PersonPositionInfo List<PersonHisInfo>

\*/

public static final String PERSONPOSITIONINFOMAP = "personPositionInfoMap";

/\*\*

\* 出差类型 id为key Map<String, TripTypeInfo> tripTypeMap

\*/

public static final String TRIPTYPEINFOMAP = "T\_HR\_ATS\_TripType";

/\*\*

\* 假期制度主表id为key Map<String, HolidayPolicySetInfo>

\*/

public static final String HOLIDAYPOLICYSETINFO = "T\_HR\_ATS\_HolidayPolicySet";

/\*\*

\* 假期类型id为key Map<String, HolidayTypeInfo>

\*/

public static final String HOLIDAYTYPEMAP = "T\_HR\_ATS\_HolidayType";

/\*\*

\* 考勤明细

\*/

public static final String ATTENDANCERESULT = "T\_HR\_ATS\_AttendanceResult";

/\*\*

\* 排班班次类型 id为key Map<String, AtsShiftTypeInfo>

\*/

public static final String ATSSHIFTTYPE = "T\_HR\_ATS\_ShiftType";

/\*\*

\* 加班原因 id为key Map<String, OverTimeReasonInfo>

\*/

public static final String OVERTIMEREASON = "T\_HR\_ATS\_OverTimeReason";

/\*\*\*

\* 工作日历明细 WorkCalendarItemInfo

\*/

public static final String WORKCALENDARITEMINFO = "WorkCalendarItemInfo" ;

/\*\*

\* 加班补偿方式 id为key Map<String, OverTimeCompensInfo>

\*/

public static final String OVERTIMECOMPENSINFO = "T\_HR\_ATS\_OverTimeCompens";

/\*\*

\* 日志主表

\*/

public static final String LOGINFO = "logInfoPk";

/\*\*

\* 日志实体

\*/

public static final String LOGINFOMAINIFNO = "logInfoMainInfo";

/\*\*

\* 是否后台事务定义触发

\*/

public static final String BACKWORKDEF ="backWorkDef";

/\*\*

\* 异常计算类

\*/

public static final String PUNCHARDEXCEPTIONPRO = "com.kingdee.shr.ats.calculate.implement.PunchardExceptionPro";

/\*\*

\* 打卡考勤请假计算类

\*/

public static final String PUNCHARDLEAVEBILLPRO = "com.kingdee.shr.ats.calculate.implement.PunchardLeaveBillPro";

/\*\*

\* 不打卡考勤请假项目计算类

\*/

public static final String NOPUNCHARDLEAVEBILLPRO = "com.kingdee.shr.ats.calculate.implement.NoPunchardLeaveBillPro";

/\*\*

\* 打卡考勤出差类项目计算类

\*/

public static final String PUNCHARDTRIPBILLPRO = "com.kingdee.shr.ats.calculate.implement.PunchardTripBillPro";

/\*\*

\*不打卡考勤出差计算类

\*/

public static final String NOPUNCHARDTRIPBILLPRO = "com.kingdee.shr.ats.calculate.implement.NoPunchardTripBillPro";

/\*\*

\* 打卡考勤加班项目计算类

\*/

public static final String PUNCHARDOVERTIMEBILLPRO = "com.kingdee.shr.ats.calculate.implement.PunchardOverTimeBillPro";

/\*\*

\* 不打卡考勤加班类项目计算类

\*/

public static final String NOPUNCHARDOVERTIMEBILLPRO = "com.kingdee.shr.ats.calculate.implement.NoPunchardOverTimeBillPro";

/\*\*

\* 打卡考勤其他项目计算类

\*/

public static final String PUNCHARDOTHERPRO = "com.kingdee.shr.ats.calculate.implement.PunchardOtherPro";

/\*\*

\* 不打卡考勤其他项目计算类

\*/

public static final String NOPUNCHARDOTHERPRO = "com.kingdee.shr.ats.calculate.implement.NoPunchardOtherPro";

/\*\*

\* fetchCardExcutor

\*/

public static final String EXCUTECALFETCHCARD = "com.kingdee.shr.ats.calculate.implement.FetchCardPro";

/\*\*

\* 考勤明细表，包含所有字段。

\*/

public static final String ATTENDANCERESULTOFALLFIELDS = "T\_HR\_ATS\_AttendanceResult";

# 附录2.考勤计算信息集速查

## 附录2.1 Map Param 信息集 结构表

|  |  |  |  |
| --- | --- | --- | --- |
| **Map param 结构** | | | |
| **键（key）** | **值说明（value description）** | **类型（type）** | **备注** |
| calculateProParamsVo | 考勤计算常用参数集合对象 | CalculateProParamsVo | 将globeMap中考勤计算过程常用参数封装成的对象避免频繁对globalMap进行检索 |
| personId | 当前计算人的id | String |  |
| attenceDate | 当前计算考勤日期（YYYY-MM-DD） | String |  |
| globalMap | 考勤计算信息总集 | Map <String,Object> | 考勤计算前准备的静态和动态信息集，包括一系列的基础数据和单据数据等 |

示例1、获取人的ID：

String personId = (String)param.get(“personId”);

示例2、获取globeMap:

Map <String,Object> globeMap = (Map <String,Object>) param.get(“globalMap”);

|  |
| --- |
| Map <String,Object> |

|  |
| --- |
| Map <String,Object> |

## 附录2.2 calculateProParamsVo 对象 结构表

**请使用getXXX/setXXX的方式对calculateProParamsVo进行读取或者修改**

|  |  |  |
| --- | --- | --- |
| **CalculateProParamsVo calculateProParamsVo 结构** | | |
| **键（key）** | **类型（value\_type）** | **值说明（value description）** |
| scheduleShiftTag | int | 几段班标识 |
| dateType | int | 日期类型 |
| hasScheduleShiftItems | boolean | 排班是否有分录 |
| workDayTag | int | 工作日标示，1代表工作日 |
| policyId | String | 考勤制度ID |
| attendTypeOne | String | 出勤类型 一/二/三 段班 |
| attendTypeTwo | String |
| attendTypeThree | String |
| oneSegmentInRest | BigDecimal | 段内休息时长 一/二/三 段班 |
| twoSegmentInRest | BigDecimal |
| threeSegmentInRest | BigDecimal |
| preFloatAdjustedone | BigDecimal | 浮动调整值 一/二/三 段班 上半段/下半段 |
| nextFloatAdjustedone | BigDecimal |
| preFloatAdjustedtwo | BigDecimal |
| nextFloatAdjustedtwo | BigDecimal |
| preFloatAdjustedthree | BigDecimal |
| nextFloatAdjustedthree | BigDecimal |
| preIsPunchCardOne | int | 是否打卡 一/二/三 段班 上半段/下半段 1代表是 |
| preIsPunchCardTwo | int |
| preIsPunchCardThree | int |
| nextIsPunchCardOne | int |
| nextIsPunchCardTwo | int |
| nextIsPunchCardThree | int |
| standardHour | BigDecimal | 标准工时，默认为8 |
| oneOnTimeDynamicResult | String | 实际取卡时间 一/二/三 段班 上半段/下半段 |
| oneOffTimeDynamicResult | String |
| twoOnTimeDynamicResult | String |
| twoOffTimeDynamicResult | String |
| threeOnTimeDynamicResult | String |
| threeOffTimeDynamicResult | String |
| oneOnTimeSchedule | String | 排班时间 一/二/三 段班 上半段/下半段 |
| oneOffTimeSchedule | String |
| twoOnTimeSchedule | String |
| twoOffTimeSchedule | String |
| threeOnTimeSchedule | String |
| threeOffTimeSchedule | String |
| oneRestPreTimeSchedule | String | 段内休息时间 一段班 上半段/下半段 |
| oneRestNextTimeSchedule | String |
| isAtLeastTwoCard | boolean | 临时加班是否需要打上下班卡 |
| otStart | int | 加班起始值 |
| otCompensIdInfo | OverTimeCompensInfo | 加班补偿方式 |
| isPreNotBill | boolean | 班前无需加班单 |
| isAfterNotBill | boolean | 班后无需加班单 |
| autoGenOverTime | boolean | 是否自动生成加班单 |
| lateAllow | int | 迟到允许值 |
| leaveAllow | int | 早退允许值 |
| absentAllow | int | 旷工起始值 |

示例1、获取迟到允许值：

int lateAllow = calculateProParamsVo.getLateAllow();

示例2、获取标准时长

BigDecimal standardHour = calculateProParamsVo.getStandardHour();

示例3、给第一段班上段班设置取卡时间

calculateProParamsVo .setOneOnTimeDynamicResult(“2016-07-01 08:00:00”);

## 附录2.3 globeMap 信息集 结构表

**请结合附录1查看此表，请使用已定义常量名对信息集进行读取。**

**考勤计算二次开发过程，请勿对信息集进行修改或者删除。**

|  |  |  |
| --- | --- | --- |
| **Map globeMap 结构** | | |
| **键（key）** | **类型（value\_type）** | **值说明（value description）** |
| **与人无关的静态信息集** | | |
| AttendanceConstants.CARDRULESINGLEINFOMAP | Map<String, CardRuleInfo> | 取卡规则单个信息的集合,取卡规则的ID作为key |
| AttendanceConstants.ATSSHIFTINFOMAP | Map<String, AtsShiftInfo> | 所有班次信息集合,班次主表ID作为key |
| AttendanceConstants.HOLIDAYPOLICYMAP | Map<String, HolidayPolicyInfo> | 所有假期制度，ID(BOSUUID)作为key |
| AttendanceConstants.ATTENDANCEPOLICYINFOMAP | Map<String, AttencePolicyInfo> | 所有考勤制度的 ID(BOSUUID)作为key |
| AttendanceConstants.OVERTIMETYPE | Map<String,OverTimeTypeInfo> | 加班类型 |
| isCalTune | boolean | 是否参与调休计算参数-公共参数 |
| isOTControl | boolean | 是否参与加班管控计算参数-公共参数 |
| isOverTimeControl | boolean | 是否参与加班管控计算参数-加班管控 |
| AttendanceConstants.WORKCALENDARINFOMAP | Map<String, WorkCalendarInfo> | 全局默认的工作日历 不打卡考勤 |
| AttendanceConstants.WORKITEMMAP | Map<String, WorkCalendarItemInfo> | 工作日历明细数据,按照开始日期和结束日期查询 |
| AttendanceConstants.TRIPTYPEINFOMAP | Map<String, TripTypeInfo> | 出差类型 |
| AttendanceConstants.HOLIDAYPOLICYSETINFO | Map<String, HolidayPolicySetInfo> | 假期制度主表信息 |
| AttendanceConstants.HOLIDAYTYPEMAP | Map<String, HolidayTypeInfo> | 假期类型 |
| AttendanceConstants.ATSSHIFTTYPE | Map<String, AtsShiftTypeInfo> | 班次类型 |
| AttendanceConstants.OVERTIMEREASON | Map<String, OverTimeReasonInfo> | 加班原因 |
| AttendanceConstants.OVERTIMECOMPENSINFO | Map<String, OverTimeCompensInfo> | 加班补偿方式 |
| **与人有关的动态信息集** | | |
| AttendanceConstants.ATTENDANCEFILEINFOMAP | Map<String, AttendanceFileInfo> | 考勤档案信息集合,人员ID作为key |
| AttendanceConstants.PPINFOMAP | Map<String,PersonPositionInfo> | 人员的 人员职位,岗位,组织,HR组织-信息集合,人员ID作为key |
| AttendanceConstants.PUNCHCARDINFOMAP | Map<String, List<PunchCardRecordInfo>> | 原始打卡记录信息,加班取卡使用, 人员ID\_考勤日期 作为key |
| AttendanceConstants.ATTENDANCERESULTINFOSMAP | Map<String, List<AttendanceDynamicResInfo>> | 人员和考勤日期对应的考勤明细信息 |
| AttendanceConstants.SCHEDULESHIFTITEMMAP | Map<String, List<ScheduleShiftInfo>> | 人员和考勤日期对应的排班分录信息,人员+考勤日期 做为key |
| AttendanceConstants.ATSLEAVEBILLINFOSMAP | Map<String, List<AtsLeaveBillEntryInfo>> | 所有请假单据信息集合,人员\_日期作为key |
| AttendanceConstants.ATSTRIPBILLINFOSMAP | Map<String, List<AtsTripBillEntryInfo>> | 所有出差单据信息集合,人员\_日期 作为key |
| AttendanceConstants.ATSOVERTIMEBILLINFOSMAP | Map<String, List<FillSignCardInfo>> | 所有加班单据信息集合,人员\_日期 作为key,以分录信息为主的集合 |
| AttendanceConstants.FILLSIGNCARDINFOSMAP | Map<String, List<FillSignCardInfo>> | 所有补签卡单据信息集合,人员\_日期作为key |
| AttendanceConstants.ATSLEAVEBILLDETAILINFOSMAP | Map<String,List<AtsLeaveBillDetailInfo>> | 请假单据明细信息集合,人员\_日期 作为key |

示例1、获取是否参与加班管控计算参数:

Boolean isOverTimeControl = (Boolean)globeMap.get(“isOverTimeControl”);

示例2、获取请假单

Map<String, List<AtsLeaveBillEntryInfo>> leaveBillMap = (Map<String, List<AtsLeaveBillEntryInfo>>)globeMap.get(AttendanceConstants.ATSLEAVEBILLINFOSMAP);

示例3、结合示例2，获取人A(ID为”abcde123”)，在2016年7月1日的所有请假单

List<AtsLeaveBillEntryInfo>> leaveList\_A = leaveBillMap.get(”abcde123”+AttendanceConstants. UNDERLINE+”2016-07-01”);