模块设计与使用文档

# 一、查询模块（未封装为step）

org.jdataex.flow.component.step.dataquery

**类 DataQuery 查询**

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** | |
| [**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) | [**getALLFSP**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\DataQuery.html#getALLFSP(java.lang.String, java.util.Map))(java.lang.String lx, java.util.Map<java.lang.String,java.lang.Object> tj)  获取非审判业务，相应系统相关表对应条件的所有记录 | |
| [**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) | [**getAllSP**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\DataQuery.html#getAllSP(java.lang.String, int))(java.lang.String ajlx, int ajbs)  获取审判业务，相关案件类型表对应案件标识的所有记录 | |
| [**RowSet**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\RowSet.html) | [**getRowSetFSP**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\DataQuery.html#getRowSetFSP(java.lang.String, java.util.Map))(java.lang.String tablename, java.util.Map<java.lang.String,java.lang.Object> conditions)  获取非审判业务，表对应条件的所有记录 | |
| [**RowSet**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\RowSet.html) | [**getRowSetSP**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\DataQuery.html#getRowSetSP(java.lang.String, int))(java.lang.String tablename, int ajbs)  获取审判业务，表对应案件标识的所有记录 | |
| java.lang.String | [**querySQLAllByAjbs**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\DataQuery.html#querySQLAllByAjbs(java.lang.String, int))(java.lang.String tablename, int ajbs)  根据案件标识获取表的所有记录，查询语句 | |
| java.lang.String | [**querySQLAllByConditions**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\DataQuery.html#querySQLAllByConditions(java.lang.String, java.lang.String[]))(java.lang.String tablename, java.lang.String[] conditionkey)  根据条件map获取表的所有记录，查询语句 | |

流程：

通过传入查询条件进行

**审判业务查询DataQuery类getAllSP(String ajlx, int ajbs)方法**:以案件类型和案件标识作为条件，读取配置文件给类型所有表，对数据库进行依次每张表查询，封装所有的字段记录信息为Row

**非审判业务查询DataQuery类getALLFSP(String lx, Map<String, Object> tj)方法**:以系统类型和相关的字段作为条件，读取配置文件给类型所有表，对数据库进行查询，封装所有的字段记录信息为Row。查询结果为该类型的所有表中含有给出的条件字段的记录（若条件有多个字段，但其中有表只含有其中一个条件字段，则该表查询条件只有该字段，含有全部字段，则查询条件为所有字段）

[**getALLSP**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\DataQuery.html#getALLFSP(java.lang.String, java.util.Map))**（ajlx，ajbs）**

获取审判业务，相关案件类型表对应案件标识的所有记录

[**getALLFSP**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\DataQuery.html#getALLFSP(java.lang.String, java.util.Map))**（lx，tj）**

**获取非审判业务记录，传入系统类型和条件字段名与值**

调用ReadDataConfig的getTablesByLX（lx）获取该类型的所有表，ajbs作为该表的查询条件字段

调用ReadDataConfig的getTablesByLX（lx）获取该类型的所有表，然后调用getColumnsByTable（tablename）获取该表的所有字段，判断是否包含条件字段，若包含，则该条件字段即为查询该表记录的条件**（若条件有多个字段，但其中有表只含有其中一个条件字段，则该表查询条件只有该字段，含有全部字段，则查询条件为所有字段）**

[**getRowSetSP**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\DataQuery.html#getRowSetSP(java.lang.String, int))(java.lang.String tablename, int ajbs)

获取审判业务，表对应案件标识的所有记录

[**getRowSetFSP**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\DataQuery.html#getRowSetFSP(java.lang.String, java.util.Map))(tablename, conditions)

获取非审判业务，表对应条件的所有记录

[**querySQLAllByAjbs**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\DataQuery.html#querySQLAllByAjbs(java.lang.String, int))(java.lang.String tablename, int ajbs)

根据案件标识获取表的所有记录，封装查询语句

querySQLAllByConditions(java.lang.String tablename, java.lang.String[] conditionkey)

根据条件map获取表的所有记录，封装查询语句

# 二、入库模块

org.jdataex.flow.component.step.dataaccess

## 类 SqlProcess 入库操作sql语句封装

|  |  |  |
| --- | --- | --- |
| **方法** | | |
| **限定符和类型** | **方法和说明** |
| java.lang.String | [**deleteSQL**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\SqlProcess.html#deleteSQL(java.lang.String, java.lang.String[]))(java.lang.String deletetable, java.lang.String[] field)  组合删除语句 |
| java.lang.String | [**getSqlList**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\SqlProcess.html#getSqlList(java.lang.String[], java.lang.String))(java.lang.String[] args, java.lang.String reg)  用于组合字符串 |
| java.lang.String | [**insertSQL**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\SqlProcess.html#insertSQL(java.lang.String, java.lang.String[]))(java.lang.String inserttable, java.lang.String[] field)  组合插入语句 |
| java.lang.String | [**updateSQL**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\SqlProcess.html#updateSQL(java.lang.String, java.lang.String[]))(java.lang.String updatetable, java.lang.String[] field)  组合更新语句 |
| java.lang.String | [**updateSQLinsert**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\SqlProcess.html#updateSQLinsert(java.lang.String, java.lang.String[], java.lang.String[]))(java.lang.String updatetable, java.lang.String[] field, java.lang.String[] con)  组合批插入时先更新语句 |

**类 RowDataAccessProcess　入库操作单条数据处理**

|  |  |  |
| --- | --- | --- |
| **方法** | | |
| **限定符和类型** | **方法和说明** |
| int | [**columnTypetoSQLType**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessProcess.html#columnTypetoSQLType(org.jdataex.flow.row.ColumnType))([**ColumnType**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\ColumnType.html) columnType)  把类型转换为JDBC类型 |
| int | [**deleteProcess**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessProcess.html#deleteProcess(java.sql.Connection, org.jdataex.flow.row.IRow))(java.sql.Connection connection, [**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row)  单条数据删除 |
| int | [**insertProcess**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessProcess.html#insertProcess(java.sql.Connection, org.jdataex.flow.row.IRow))(java.sql.Connection connection, [**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row)  单条数据插入 |
| int | [**updateProcess**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessProcess.html#updateProcess(java.sql.Connection, org.jdataex.flow.row.IRow))(java.sql.Connection connection, [**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row)  单条数据更新 |

org.jdataex.flow.component.step.dataaccess

## 类 RowDataAccess　数据入库事务处理

### 构造器

#### RowDataAccess

public RowDataAccess(java.lang.String poolname)

**参数:**

poolname - 连接池名

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| void | [**doDelete**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccess.html#doDelete(org.jdataex.flow.row.IRow))([**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row)  批数据删除事务 |
| void | [**doInsert**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccess.html#doInsert(org.jdataex.flow.row.IRow))**\_Update**([**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row)  批数据插入事务(先插入再尝试更新) |
| void | [**do**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccess.html#doInsert(org.jdataex.flow.row.IRow))**Update\_Insert** ([**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row)  批数据插入事务(先更新,若为0再插入) |
| void | [**doProcessRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccess.html#doProcessRow(org.jdataex.flow.row.IRow))([**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row)  批处理数据 |

org.jdataex.flow.component.step.dataaccess

## 类 RowDataAccessStep　封装入库的组件

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| void | [**afterBatchProcessing**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#afterBatchProcessing())()  在处理数据后，需要处理的事情。 |
| void | [**afterTurnOff**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#afterTurnOff())()  在组件当次运行后，需要处理的事情。 |
| void | [**beforeBatchProcessing**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#beforeBatchProcessing())()  在处理数据前，需要处理的事情。 |
| void | [**beforeTurnOn**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#beforeTurnOn())()  在组件当次运行前，需要处理的事情。 |
| void | [**beginRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#beginRow(org.jdataex.flow.row.IRow))([**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row)  在处理当前一条数据前，需要处理的事情。 |
| void | [**beginRowSet**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#beginRowSet())()  在处理当前一批数据前，需要处理的事情。 |
| void | [**commitRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#commitRow(org.jdataex.flow.row.IRow))([**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row)  在处理当前一条数据成功后，需要处理的事情。 |
| void | [**commitRowSet**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#commitRowSet())()  在处理当前一批数据成功后，需要处理的事情。 |
| void | [**init**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#init())() |
| void | [**initConfig**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#initConfig(org.jdataex.engine.config.ComponentItem))([**ComponentItem**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\engine\config\ComponentItem.html) item)  只用于初始化组件所用到的参数。 |
| void | [**processRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#processRow(org.jdataex.flow.row.IRow))([**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row) |
| void | [**release**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#release())() |
| void | [**rollbackRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#rollbackRow(org.jdataex.flow.row.IRow))([**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row)  在处理当前一条数据失败后，需要处理的事情。 |
| void | [**rollbackRowSet**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataaccess\RowDataAccessStep.html#rollbackRowSet())()  在处理当前一批数据失败后，需要处理的事情。 |

## 流程：

框架传递XML文件封装好的Row，RowDataAccessStep初始化开始处理，调用RowDataAccess

的doProcess处理，根据封装的Row的ＡＣＴＩＯＮ＿ＴＹＰＥ分别调用删除和插入方法进行事务处理（目前insert为插入（先insert再update），delete为删除，其他的则为插入（先update再insert））

**插入操作**：一个XMl文件为一个事务，即一个Row，每个Row里的每条表的记录为一个数据库操作：

**默认**：**insert\_update**:先insert，若出错，则尝试update，若update出错或者返回值为0，则中断本次插入事务进行回滚操作

**其他：update\_insert:**插入前，先以传递过来的该表的字段数据为条件（**除去blob字段和空字段**），做update操作（update除ID和REG\_TIME外的所有字段），若update出错或者update列数为0，则回滚该update操作并对该条记录进行insert操作。

整个事务中有一条insert出错，则整个事务的所有操作进行回滚处理。

**删除操作**：一个XMl文件为一个事务，即一个Row，每个Row里的每条表的记录为一个数据库操作：以传递过来的该表的字段数据为条件（**除去blob字段和空字段**），做delete操作，若一条delete出错，则整个事务的所有操作进行回滚处理，若delete的列数为0，做记录。

init()初始化该组件的配置

RowDataAccessStep

processRow（）处理row

RowDataAccess

doProcessRow（）

doInsert\_update（）

doDelete（）

doUpdate\_insert

RowDataAccessProcess

insertProcess()

deleteProcess（）

updateProcess（）

updateProcess()

insertProcess（）

# 三、数据配置文件（数据库表字段等配置）读取

org.jdataex.flow.component.step.dataquery

## 类 ReadDataConfig　读取数据配置文件

|  |  |  |
| --- | --- | --- |
| **方法** | | |
| **限定符和类型** | **方法和说明** |
| java.lang.String | [**getBSByLX**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\ReadDataConfig.html#getBSByLX(java.lang.String))(java.lang.String lx)  根据类型获取标识 |
| java.util.Map<java.lang.String,java.lang.String> | [**getColumnAndTypesByTable**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\ReadDataConfig.html#getColumnAndTypesByTable(java.lang.String))(java.lang.String tablename)  根据表名获取字段名和字段类型 |
| java.util.List<java.lang.String> | [**getColumnsByTable**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\ReadDataConfig.html#getColumnsByTable(java.lang.String))(java.lang.String tablename)  根据表名获取要查询的字段名 |
| java.util.List<java.lang.String> | [**getConditionColumnByTable**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\ReadDataConfig.html#getConditionColumnByTable(java.lang.String))(java.lang.String tablename)  根据表名获取条件字段 |
| static [**ReadDataConfig**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\ReadDataConfig.html) | [**getInstance**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\ReadDataConfig.html#getInstance())() |
| java.util.List<java.lang.String> | [**getTablesByLX**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\ReadDataConfig.html#getTablesByLX(java.lang.String))(java.lang.String lx)  根据类型获取表名 |
| private void | [**load**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\dataquery\ReadDataConfig.html#load())()  迭代读取数据配置文件，加载表对应字段关系，类型对应的标识，类型对应的表 |

调用jdom的api读取配置上文件，配置文件格式如下：

<SPYW>

<LX name="类型名（非审判业务为系统类型，审判业务为案件类型）" bz="含有公共字段的标识（如审判业务的ＡＪＢＳ）">

<table name="表名">

<column name="字段名" type="字段类型" flag="是否作为条件"/>

<column name="XH" type="int" flag="true"/>

<column name="XFR" type="int" flag="true"/>

<column name="XFRSF" type="int" flag="true"/>

<column name="YDSRGX" type="String" flag="true"/>

<column name="XWNL" type="int" flag="true"/>

</table>

</LX>

<SPYW>

# 四、触发器获取模块

org.jdataex.flow.component.step.trigger

**类 ConfigBean　　封装触发器配置**

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| static [**ConfigBean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\ConfigBean.html) | [**getInstance**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\ConfigBean.html#getInstance())()  触发器配置实例 |
| [**TriggerDataBean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html) | [**getTriggerDataBean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\ConfigBean.html#getTriggerDataBean(org.jdataex.flow.component.step.trigger.TriggerDataBean))([**TriggerDataBean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html) triggerDataBean)  获取触发器配置信息 |
| private void | [**load**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\ConfigBean.html#load())()  加载配置 |

org.jdataex.flow.component.step.trigger

**类 DataHandler　数据处理类（主要是触发器相关的中间表，源表数据获取等）**

|  |  |
| --- | --- |
| **构造器** | |
| **构造器和说明** | |
| [**DataHandler**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DataHandler.html#DataHandler(org.jdataex.flow.component.step.trigger.TriggerDataBean, java.lang.String))([**TriggerDataBean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html) triggerDataBean, java.lang.String poolname) | |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| [**TriggerDataBean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html) | [**getTriggerDataBean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DataHandler.html#getTriggerDataBean())()  加载配置文件根据事务名获取配置中的bean内容 |
| [**TriggerDataBean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html) | [**middleDataGet**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DataHandler.html#middleDataGet(int, java.lang.String))(int msgid, java.lang.String tr\_msg)  获取当前中间表记录的表对应关系 |
| void | [**setCode**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DataHandler.html#setCode())()  预留用于设置app\_code和dept\_code; |
| void | [**souDataGet**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DataHandler.html#souDataGet(org.jdataex.flow.component.step.trigger.TriggerDataBean))([**TriggerDataBean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html) triggerDataBean)  数据库源表数据值获取 |
| [**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) | [**SouToDest**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DataHandler.html#SouToDest())()  源表数据转换为目标数据 |
| void | [**updateMsgid**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DataHandler.html#updateMsgid(int, java.lang.String))(int msgid, java.lang.String field)  更新msgid |

org.jdataex.flow.component.step.trigger

**类 DestData　封装触发器获取目前数据的类**

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| java.lang.String | [**getAction**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DestData.html#getAction())() |
| java.util.Map<java.lang.String,java.lang.String> | [**getField\_type**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DestData.html#getField_type())() |
| java.util.Map<java.lang.String,java.lang.Object> | [**getField\_value**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DestData.html#getField_value())() |
| java.lang.String | [**getTable**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DestData.html#getTable())() |
| void | [**setAction**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DestData.html#setAction(java.lang.String))(java.lang.String action) |
| void | [**setField\_type**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DestData.html#setField_type(java.util.Map))(java.util.Map<java.lang.String,java.lang.String> field\_type) |
| void | [**setField\_value**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DestData.html#setField_value(java.util.Map))(java.util.Map<java.lang.String,java.lang.Object> field\_value) |
| void | [**setTable**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\DestData.html#setTable(java.lang.String))(java.lang.String table) |

org.jdataex.flow.component.step.trigger

**类 ExcelConfigLoad　触发器配置excel文件载入处理类**

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| void | [**createConfig**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\ExcelConfigLoad.html#createConfig(java.lang.String, java.lang.String))(java.lang.String currentpath, java.lang.String configpath)  更改配置文件（新增触发器配置） |
| void | [**deleteConfig**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\ExcelConfigLoad.html#deleteConfig(java.lang.String, java.lang.String))(java.lang.String currentpath, java.lang.String configpath)  更改配置文件（删除触发器配置） |
| java.util.List<java.lang.Integer> | [**getLocation**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\ExcelConfigLoad.html#getLocation())() |
| java.util.Map<java.lang.String,[**TriggerDataBean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html)> | [**getServ\_bean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\ExcelConfigLoad.html#getServ_bean())() |
| java.util.List<java.lang.String> | [**getServiceList**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\ExcelConfigLoad.html#getServiceList())() |
| java.util.List<java.lang.String> | [**getTable**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\ExcelConfigLoad.html#getTable())() |
| java.util.List<[**TriggerDataBean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html)> | [**getTriggerDataBeanList**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\ExcelConfigLoad.html#getTriggerDataBeanList())() |
| void | [**loadConfig**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\ExcelConfigLoad.html#loadConfig(java.lang.String))(java.lang.String path)  加载配置文件 |
| void | [**updateConfig**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\ExcelConfigLoad.html#updateConfig(java.lang.String, java.lang.String))(java.lang.String currentpath, java.lang.String configpath)  更改配置文件（更改触发器配置） |

org.jdataex.flow.component.step.trigger

**类 TriggerComponent**

**触发器组件相关（此类需修改）**

|  |  |
| --- | --- |
| **构造器** | |
| **构造器和说明** | |
| [**TriggerComponent**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerComponent.html#TriggerComponent(int, int, java.lang.String, java.lang.String))(int threadnum, int initthreadnum, java.lang.String datatype, java.lang.String poolname) | |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| void | [**configHandler**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerComponent.html#configHandler())()  触发器配置处理 |
| void | [**dataHandler**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerComponent.html#dataHandler())() |
| boolean | [**deleteFile**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerComponent.html#deleteFile(java.lang.String))(java.lang.String sPath)  删除单个文件 |
| boolean | [**fileMove**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerComponent.html#fileMove(java.lang.String, java.lang.String))(java.lang.String source, java.lang.String dest) |

org.jdataex.flow.component.step.trigger

**类 TriggerDataBean　　用于封装触发器同步配置相关数据的Bean**

|  |
| --- |
| **构造器** |
| **构造器和说明** |
| [**TriggerDataBean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#TriggerDataBean())() |
| [**TriggerDataBean**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#TriggerDataBean(java.lang.String))(java.lang.String poolname) |

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| java.lang.String | [**getAction**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getAction())()  获取源操作 |
| java.lang.String | [**getDestaction**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getDestaction())()  获取目的操作 |
| java.util.List<java.lang.String> | [**getDestfield**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getDestfield())()  获取目的字段 |
| java.lang.String | [**getDesttablename**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getDesttablename())()  获取目的表名 |
| java.util.List<java.lang.String> | [**getDesttype**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getDesttype())()  获取目的字段类型列表 |
| java.util.List<java.lang.String> | [**getExchanges**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getExchanges())()  获取转换表达式 |
| java.util.List<java.lang.String> | [**getField**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getField())()  获取源字段 |
| java.lang.String[] | [**getMsgparam**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getMsgparam())()  获取中间表值 |
| java.util.List<java.lang.String> | [**getPrimarykey**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getPrimarykey())()  获取主键 |
| java.lang.String | [**getService**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getService())()  获取事务名 |
| int | [**getServiceid**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getServiceid())()  获取事务ID |
| java.util.Map<java.lang.String,java.lang.String> | [**getSourcetype**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getSourcetype())()  获取源字段对应类型 |
| java.util.Map<java.lang.String,java.lang.Object> | [**getSourcevalue**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getSourcevalue())()  获取源字段对应值 |
| java.lang.String | [**getSql**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getSql())() |
| java.lang.String | [**getTablename**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getTablename())()  获取表名 |
| java.util.List<java.lang.String> | [**getType**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#getType())()  获取源字段类型列表 |
| void | [**setAction**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setAction(java.lang.String))(java.lang.String action) |
| void | [**setDestaction**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setDestaction(java.lang.String))(java.lang.String destaction) |
| void | [**setDestfield**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setDestfield(java.util.List))(java.util.List<java.lang.String> destfield) |
| void | [**setDesttablename**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setDesttablename(java.lang.String))(java.lang.String desttablename) |
| void | [**setDesttype**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setDesttype(java.util.List))(java.util.List<java.lang.String> desttype) |
| void | [**setExchanges**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setExchanges(java.util.List))(java.util.List<java.lang.String> exchanges) |
| void | [**setField**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setField(java.util.List))(java.util.List<java.lang.String> field) |
| void | [**setMsgparam**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setMsgparam(java.lang.String[]))(java.lang.String[] msgparam) |
| void | [**setPrimarykey**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setPrimarykey())() |
| void | [**setService**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setService(java.lang.String))(java.lang.String service) |
| void | [**setServiceid**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setServiceid(int))(int serviceid) |
| void | [**setSourcetype**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setSourcetype(java.util.Map))(java.util.Map<java.lang.String,java.lang.String> sourcetype) |
| void | [**setSourcevalue**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setSourcevalue(java.util.Map))(java.util.Map<java.lang.String,java.lang.Object> sourcevalue) |
| void | [**setSql**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setSql(java.lang.String))(java.lang.String sql) |
| void | [**setTablename**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setTablename(java.lang.String))(java.lang.String tablename) |
| void | [**setType**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataBean.html#setType(java.util.List))(java.util.List<java.lang.String> type) |

org.jdataex.flow.component.step.trigger

**类 TriggerDataGetStep　触发器获取组件**

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| void | [**afterBatchProcessing**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#afterBatchProcessing())()  在处理数据后，需要处理的事情。 |
| void | [**afterTurnOff**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#afterTurnOff())()  在组件当次运行后，需要处理的事情。 |
| void | [**beforeBatchProcessing**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#beforeBatchProcessing())()  在处理数据前，需要处理的事情。 |
| void | [**beforeTurnOn**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#beforeTurnOn())()  在组件当次运行前，需要处理的事情。 |
| void | [**beginRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#beginRow(org.jdataex.flow.row.IRow))([**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row)  在处理当前一条数据前，需要处理的事情。 |
| void | [**beginRowSet**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#beginRowSet())()  在处理当前一批数据前，需要处理的事情。 |
| void | [**commitRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#commitRow(org.jdataex.flow.row.IRow))([**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row)  在处理当前一条数据成功后，需要处理的事情。 |
| void | [**commitRowSet**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#commitRowSet())()  在处理当前一批数据成功后，需要处理的事情。 |
| void | [**init**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#init())() |
| void | [**initConfig**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#initConfig(org.jdataex.engine.config.ComponentItem))([**ComponentItem**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\engine\config\ComponentItem.html) item)  只用于初始化组件所用到的参数。 |
| void | [**processRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#processRow(org.jdataex.flow.row.IRow))([**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row) |
| void | [**release**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#release())() |
| void | [**rollbackRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#rollbackRow(org.jdataex.flow.row.IRow))([**IRow**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\row\IRow.html) row)  在处理当前一条数据失败后，需要处理的事情。 |
| void | [**rollbackRowSet**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#rollbackRowSet())()  在处理当前一批数据失败后，需要处理的事情。 |
| void | [**setMsgid**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerDataGetStep.html#setMsgid(java.lang.String))(java.lang.String loc) |

org.jdataex.flow.component.step.trigger

**类 TriggerSqlHandler　触发器sql组合执行**

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| void | [**triggerCreateOrReplaceSql**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerSqlHandler.html#triggerCreateOrReplaceSql(java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String[]))(java.lang.String tablename, java.lang.String triggername, java.lang.String action, java.lang.String middletable, java.lang.String[] primarykey)  oracle创建触发器 |
| void | [**triggerCreateOrReplaceSqlSybase**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerSqlHandler.html#triggerCreateOrReplaceSqlSybase(java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String[]))(java.lang.String tablename, java.lang.String triggername, java.lang.String action, java.lang.String middletable, java.lang.String[] primarykey)  Sybase创建触发器 |
| void | [**triggerDeleteSql**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerSqlHandler.html#triggerDeleteSql(java.lang.String, java.lang.String))(java.lang.String triggername, java.lang.String tablename)  删除触发器 |
| void | [**triggerDeleteSqlSybase**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerSqlHandler.html#triggerDeleteSqlSybase(java.lang.String, java.lang.String))(java.lang.String triggername, java.lang.String tablename)  Sybase删除触发器 |
| void | [**updateTrigger**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerSqlHandler.html#updateTrigger(java.lang.String, java.lang.String, java.lang.String, java.lang.String[]))(java.lang.String tablename, java.lang.String triggername, java.lang.String action, java.lang.String[] primarykey)  更新触发器 |
| void | [**updateTriggerSybase**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\step\trigger\TriggerSqlHandler.html#updateTriggerSybase(java.lang.String, java.lang.String, java.lang.String, java.lang.String[]))(java.lang.String tablename, java.lang.String triggername, java.lang.String action, java.lang.String[] primarykey)  Sybase 更新触发器 |

## 相关说明：

**准备：数据库，用户需要授权查找v$session表，供创建的触发器判断操作用户**

触发器需创建如下表：（中间表和Tr\_msgids需根据线程数提前创建）

Tr\_Trigger

(

MsgServiceName varchar(255),//服务名

MsgTableName varchar(255),//表名

MsgTriggerName varchar(255)//触发器名

MsgMiddleName varchar(255)

)

Tr\_Msg0

(

MsgServiceName varchar(255),//服务

MsgSourceTableName varchar(255) NULL,//记录来自表名

MsgInsertAt datetime,//记录插入时间

MsgAction varchar(10),//记录动作

MsgParam1 varchar(255) NULL,

MsgParam2 varchar(255) NULL,

MsgParam3 varchar(255) NULL,

MsgParam4 varchar(255) NULL,

MsgParam5 varchar(255) NULL,

MsgId numeric(32)   IDENTITY NOT NULL PRIMARY KEY ,//oracle需用序列和触发器

)

上表的数目根据可能需要的最大线程数创建

Tr\_Msgids(

Tr\_Msg0 Integer;//根据线程数创建字段数，初始设置值为0，对应中间表取的处理的msgids的值

)

**其中oracle数据库，还得为每个中间表Tr\_msg创建sequence和id的触发器使其能够自增长，sql语句如下**

建立序列：create sequence seq\_trmsg0 increment by **1** start with **1**

create or replace trigger tr\_seq\_trmsg0 before insert on Tr\_Msg0 for each row

begin select seq\_trmsg0.nextval into :new.msgid from dual; end;

EXCEL文件配置格式：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Servicename | sourcetablename | action | sourcefield | desttablename | destaction | destfield | desttype | Exchanges |
| 服务名  ：即一个事务，触发器名根据这个创建，若为空，则该名与源表名相同 | 源表名：  不能为空 | 触发动作：不为空  0：所有  1：insert  2：update  3：delete  4：insert、update | 源表字段：若为空，则默认为所有字段，同时若该表没有主键，则这里必须有一个值默认为主键 | 目的表名：若为空，则表名与源表相同 | 目的动作，若为空则与源表所触发的动作相同 | 目的字段：  若为空,与源表字段相同 | 目的字段类型：  若为空则与源表字段相同 | 转换：  转换公式需符合avitor，同时与字段必须对应 |

**配置说明：**

触发器，表映射关系配置目录分为两个：配置更新目录（updateconfig）与当前配置目录（currentconfig）

updateconfig目录放置excel文件，其中新建（create），更改（update），删除（delete）**表**映射关系（即需要产生触发器）各一个文件，文件内容模板相同，当系统启动时或进行手动配置获取，若updateconfig目录下有文件，则读取对应的文件

1：读取create文件，则新建触发器，同时，将文件中的内容插入到currentconfig目录下的current文件（若current下没有该文件则复制文件），同时删除updateconfig目录下的文件。

2：读取update文件，则修改对应的触发器，同时文件中的内容与currentconfig下的current文件做对应修改后删除文件。

3：读取delete文件，删除对应的触发器，同时删除currentconfig下的current文件中对应内容后删除文件。

**触发器创建：（这个代码需要更改为创建触发器时，同时根据线程创建的中间表数，生成触发器获取的step组件的配置文件，以使用框架的多线程，每个触发器获取的线程对应一张中间表）**

第一次创建，读取updateconfig目录下的create文件，根据线程数创建中间数据表，线程与中间表一一对应，每个线程依次读取，创建触发器。

后续新建，根据当前的中间表的数与线程数，做比较，若小于则创建少的几张表，若大于，则建触发器不变化，在数据获取时，单独运行线程来对另外的几张表的处理。

触发器为插入源表的主键值，若该表无主键，则默认选择配置文件第一个字段为主键，若都没有，则进行报错等相关信息提示。

每一个service对应一张源表，源表上的每个动作（insert，update，delete）对应一个触发器，触发器名字为该service名+动作名，service\_insert。同时生成触发器获取组件的配置文件中调用触发器组件的部分，配置参考，一张中间表对应如下一块配置

<component id="triggerget1"

class="org.jdataex.flow.component.step.trigger.TriggerDataGetStep">

<station-ref type="input" id="sid1"/>

<station-ref type="output" id="sid2"/>

<pool-name>pool</pool-name>

<table-name>Tr\_Msg1</table-name>

</component>

**数据获取处理流程说明：**

TriggerDataGetStep的processRow根据配置文件从数据库表Tr\_Msgids中，获取对应先前最后处理的中间表数据序列号位置MSGID，DataHandler的middleDataGet(msgid + 1, tablename)获取中间表的下一条记录，DataHandler的souDataGet(triggerDataBean)读取源表的记录，根据service名去映射配置文件中查找对应的映射关系DataHandler的SouToDest()根据关系对源表记录数据进行转换，封装为目的表的数据（每配置上述触发器获取组件的模块一个，则对应一个线程处理一个中间表），依次msgid+1进行，直到中间表没有msgid大于Tr\_msgids表中对应的msgid值。（同时返回的Row，还应该封装上ＡＰＰ＿ＣＯＤＥ和ＤＥＰＴ＿ＣＯＤＥ）

# 五、数据库连接池

org.jdataex.resource.databasepool

**类 ConfigDatabasePoolBean　封装数据库连接池配置信息的Bean**

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| java.lang.String | [**getDatabaseType**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#getDatabaseType())()  数据库类型 |
| java.lang.String | [**getDbPassword**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#getDbPassword())()  密码 |
| java.lang.String | [**getDbUrl**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#getDbUrl())()  地址 |
| java.lang.String | [**getDbUsername**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#getDbUsername())()  用户名 |
| int | [**getIncrConnections**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#getIncrConnections())()  初始连接数 |
| java.lang.String | [**getJdbcDriver**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#getJdbcDriver())()   驱动 |
| int | [**getMaxConnections**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#getMaxConnections())()  最大连接数 |
| int | [**getMinConnections**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#getMinConnections())()  最小连接数 |
| java.lang.String | [**getPoolname**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#getPoolname())()  连接池名 |
| void | [**setDatabaseType**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#setDatabaseType(java.lang.String))(java.lang.String databaseType) |
| void | [**setDbPassword**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#setDbPassword(java.lang.String))(java.lang.String dbPassword) |
| void | [**setDbUrl**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#setDbUrl(java.lang.String))(java.lang.String dbUrl) |
| void | [**setDbUsername**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#setDbUsername(java.lang.String))(java.lang.String dbUsername) |
| void | [**setIncrConnections**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#setIncrConnections(int))(int incrConnections) |
| void | [**setJdbcDriver**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#setJdbcDriver(java.lang.String))(java.lang.String jdbcDriver) |
| void | [**setMaxConnections**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#setMaxConnections(int))(int maxConnections) |
| void | [**setMinConnections**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#setMinConnections(int))(int minConnections) |
| void | [**setPoolname**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ConfigDatabasePoolBean.html#setPoolname(java.lang.String))(java.lang.String poolname) |

org.jdataex.resource.databasepool

**类 ParseDSConfig　数据库连接池配置文件读取**

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| java.util.Vector | [**readConfigInfo**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\ParseDSConfig.html#readConfigInfo(java.lang.String))(java.lang.String path)  读取xml配置文件 |

连接池配置文件格式如下：

<ds-config>

<pool>

<poolname>poolxxx</poolname><!—-连接池名-->

<jdbcdriver>oracle.jdbc.driver.OracleDriver</jdbcdriver><!—-JDBC驱动-->

<databasetype>oracle</databasetype><!—-数据库类型-->

<dburl>jdbc:oracle:thin:@10.10.22.12:1521:ORCL</dburl><!—-数据库URL地址-->

<dbusername>sgy</dbusername><!—-数据库用户名-->

<dbpassword>sgy</dbpassword><!—-数据库密码-->

<minconnections>1</minconnections><!—-连接池最小连接数-->

<maxconnections>100</maxconnections><!—-连接池最大连接数-->

</pool>

<pool>

<poolname>pool</poolname>

<jdbcdriver>oracle.jdbc.driver.OracleDriver</jdbcdriver>

<databasetype>oracle</databasetype>

<dburl>jdbc:oracle:thin:@10.10.22.12:1521:ORCL</dburl>

<dbusername>sgy</dbusername>

<dbpassword>sgy</dbpassword>

<minconnections>1</minconnections>

<maxconnections>100</maxconnections>

</pool>

</dsconfig>

org.jdataex.resource.databasepool

**类 DBManager　　连接池管理类（封装了ＢｏｎｅＣＰ连接池）**

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| void | [**close**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\DBManager.html#close())()  关闭连接池 |
| java.sql.Connection | [**getConnection**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\DBManager.html#getConnection(java.lang.String))(java.lang.String poolname)  获取指定连接池名的连接 |
| static [**DBManager**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\DBManager.html) | [**getInstance**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\DBManager.html#getInstance())()  获取连接池管理实例 |
| private void | [**init**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\DBManager.html#init())()  连接池初始化 |
| private void | [**loadConfig**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\DBManager.html#loadConfig())()  加载连接池配置文件 |
| static void | [**setPath**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\resource\databasepool\DBManager.html#setPath(java.lang.String))(java.lang.String path)  设置配置文件路径 |

org.jdataex.flow.component

**类 DBOperation　　数据库常用操作封装**

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| java.lang.String[] | [**getTablePK**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\DBOperation.html#getTablePK(java.lang.String, java.lang.String))(java.lang.String table, java.lang.String poolname)  指定连接池，获取表主键 |
| boolean | [**sqlExecute**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\DBOperation.html#sqlExecute(java.lang.String, java.lang.String))(java.lang.String sql, java.lang.String poolname)  指定连接池，执行SQL语句 |
| java.util.List | [**sqlQuery**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\DBOperation.html#sqlQuery(java.lang.String, java.lang.Object[], int[], java.lang.String))(java.lang.String sql, java.lang.Object[] param, int[] type, java.lang.String poolname)  指定连接池，执行查询SQL语句 |
| java.sql.ResultSet | [**sqlQuery**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\DBOperation.html#sqlQuery(java.lang.String, java.lang.Object[], java.lang.String))(java.lang.String sql, java.lang.Object[] param, java.lang.String poolname)  指定连接池，执行查询SQL语句 |
| java.sql.ResultSet | [**sqlQuery**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\DBOperation.html#sqlQuery(java.lang.String, java.lang.String))(java.lang.String sql, java.lang.String poolname)  指定连接池，执行查询SQL语句 |
| int | [**sqlStringUpdate**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\DBOperation.html#sqlStringUpdate(java.lang.String, java.lang.String[], java.lang.String))(java.lang.String sql, java.lang.String[] param, java.lang.String poolname)  指定连接池，执行SQL语句，占位符方式，默认类型String |
| int | [**sqlUpdate**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\DBOperation.html#sqlUpdate(java.lang.String, java.lang.Object[], int[], java.lang.String))(java.lang.String sql, java.lang.Object[] param, int[] type, java.lang.String poolname)  指定连接池，执行SQL语句，占位符方式 |
| int | [**sqlUpdate**](file:///G:\code\j2se\workspace\JDataEx\org\jdataex\flow\component\DBOperation.html#sqlUpdate(java.lang.String, java.lang.Object[], java.lang.String))(java.lang.String sql, java.lang.Object[] param, java.lang.String poolname)  指定连接池，执行SQL语句，占位符方式，不指定类型 |

# 六、API接口

API接口需用到监控程序保存消息的库表为

create table MESSAGES(

uuid VARCHAR2(36) not null,UUID，消息唯一识别码

destappcode VARCHAR2(32) not null,目标应用编码

destdeptcode VARCHAR2(32) not null,目标法院编码

sourceappcode VARCHAR2(32) not null,源应用编码

sourcedeptcode VARCHAR2(32) not null,源法院编码

msgtype INTEGER not null, 消息类型（0：请求消息，1：数据消息，2：应急请求消息，3：应急数据消息）

msgdata VARCHAR2(255),buf消息内容（btye[] 类型直接通过 setString（“msgdata”，new String（msgdata.getBytes（））插入，否则接口查询会出现问题））

filepath VARCHAR2(255),文件消息的文件路径

attrs VARCHAR2(255),用户自定义属性格式为（属性名1:属性值1;属性名2:属性值2 如：name:xiaoming;age: 15）

inserttime TIMESTAMP(6) 插入表的时间

)

API接口中面向第三方的主要是包 org.jdataex.message，其中的类详见文件

org.jdataex.message

**类 MessageExceptions 对异常进行封装**

|  |
| --- |
| **构造器** |
| [**MessageExceptions**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\message\MessageExceptions.html#MessageExceptions())() |
| [**MessageExceptions**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\message\MessageExceptions.html#MessageExceptions(java.lang.String))(java.lang.String message) |
| [**MessageExceptions**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\message\MessageExceptions.html#MessageExceptions(java.lang.String, java.lang.Throwable))(java.lang.String message, java.lang.Throwable cause) |
| [**MessageExceptions**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\message\MessageExceptions.html#MessageExceptions(java.lang.String, java.lang.Throwable, boolean, boolean))(java.lang.String message, java.lang.Throwable cause, boolean enableSuppression, boolean writableStackTrace) |
| [**MessageExceptions**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\message\MessageExceptions.html#MessageExceptions(java.lang.Throwable))(java.lang.Throwable cause) |

其他类:

org.jdataex.tlq

**类 BaseTLQ TLQ的基本参数封装**

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| protected com.tongtech.tlq.base.TlqQCU | [**getQcu**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\BaseTLQ.html#getQcu())()  获取qcu |
| java.lang.String | [**getQcuName**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\BaseTLQ.html#getQcuName())()  获取qcu名 |
| java.lang.String | [**getQueueName**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\BaseTLQ.html#getQueueName())()  获取队列名 |
| int | [**getTimes**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\BaseTLQ.html#getTimes())()  获取重试次数 |
| protected com.tongtech.tlq.base.TlqConnection | [**getTlqConnection**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\BaseTLQ.html#getTlqConnection())()  获取tlq连接 |
| int | [**getType**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\BaseTLQ.html#getType())()  获取消息类型 |
| protected void | [**setQcu**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\BaseTLQ.html#setQcu(com.tongtech.tlq.base.TlqQCU))(com.tongtech.tlq.base.TlqQCU qcu) |
| void | [**setQcuName**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\BaseTLQ.html#setQcuName(java.lang.String))(java.lang.String qcuName) |
| void | [**setQueueName**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\BaseTLQ.html#setQueueName(java.lang.String))(java.lang.String queueName) |
| void | [**setTimes**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\BaseTLQ.html#setTimes(int))(int times) |
| protected void | [**setTlqConnection**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\BaseTLQ.html#setTlqConnection(com.tongtech.tlq.base.TlqConnection))(com.tongtech.tlq.base.TlqConnection tlqConnection) |
| void | [**setType**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\BaseTLQ.html#setType(int))(int type) |

org.jdataex.tlq

**类 PutTLQ ＴＬＱ操作封装继承ＢａｓｅＴＬＱ**

|  |
| --- |
| **构造器** |
| **构造器和说明** |
| [**PutTLQ**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#PutTLQ(java.lang.String, java.lang.String, boolean, java.lang.String, int))(java.lang.String qcuName, java.lang.String queueName, boolean autoReConnect, java.lang.String hostName, int hostPort)  传入ｑｃｕ名，队列名，连接失败是否重新连接，ｔｌｑ节点地址，ｔｌｑ节点端口 |

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| void | [**close**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#close())() |
| void | [**connect**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#connect())()  连接本地的tlq或者远程的tlq,建议一般情况下连接本地的tlq 进行消息发送,如果是本地 的tlq,hostName直接传入 null |
| private void | [**connectOnce**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#connectOnce())() |
| java.lang.String | [**getHostName**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#getHostName())() |
| int | [**getHostPort**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#getHostPort())() |
| boolean | [**isAutoReConnect**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#isAutoReConnect())() |
| void | [**putBuffMsg**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#putBuffMsg(byte[], org.jdataex.message.MessageAttrBean))(byte[] msgContent, [**MessageAttrBean**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\message\MessageAttrBean.html) messageAttrs)  发送ｂｕｆ消息 |
| void | [**putFileMsg**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#putFileMsg(java.lang.String, org.jdataex.message.MessageAttrBean))(java.lang.String file, [**MessageAttrBean**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\message\MessageAttrBean.html) messageAttrs)  发送文件消息 |
| private void | [**reConnect**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#reConnect())()  错误不捕获,直接抛出,意味着如果TLQ环境本身就有问题将不会重试 |
| void | [**run**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#run())()  自动重连tlqConnect,如果遇到TLQ环境错误等类型的问题,直接关闭重试 首先判断连接是否正常是否能发生消息如果不能才重连; |
| void | [**setAutoReConnect**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#setAutoReConnect(boolean))(boolean autoReConnect) |
| void | [**setHostName**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#setHostName(java.lang.String))(java.lang.String hostName) |
| void | [**setHostPort**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#setHostPort(int))(int hostPort) |
| private void | [**testAutoReconnectMsg**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\tlq\PutTLQ.html#testAutoReconnectMsg())() |

org.jdataex.common

**类 AssertUtil 断言一些常用的值**

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| static void | [**assertFalse**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\AssertUtil.html#assertFalse(java.lang.String, boolean))(java.lang.String msg, boolean value) |
| static void | [**assertLessThanNumber**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\AssertUtil.html#assertLessThanNumber(java.lang.String, int, int))(java.lang.String msg, int number, int consult) |
| static void | [**assertNegativeInt**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\AssertUtil.html#assertNegativeInt(int))(int number) |
| static void | [**assertNegativeInt**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\AssertUtil.html#assertNegativeInt(java.lang.String, int))(java.lang.String msg, int number) |
| static void | [**assertNotClass**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\AssertUtil.html#assertNotClass(java.lang.Object, java.lang.Class))(java.lang.Object obj, java.lang.Class clasz) |
| static void | [**assertNull**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\AssertUtil.html#assertNull(java.lang.Object))(java.lang.Object object) |
| static void | [**assertNull**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\AssertUtil.html#assertNull(java.lang.String, java.lang.Object))(java.lang.String message, java.lang.Object object) |
| static void | [**assertNullOrBlank**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\AssertUtil.html#assertNullOrBlank(java.lang.String))(java.lang.String text) |
| static void | [**assertNullOrBlank**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\AssertUtil.html#assertNullOrBlank(java.lang.String, java.lang.String))(java.lang.String message, java.lang.String text) |
| static void | [**assertNullOrBlankSingleMessage**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\AssertUtil.html#assertNullOrBlankSingleMessage(java.lang.String, java.lang.String))(java.lang.String partMessage, java.lang.String text) |
| static void | [**assertNullSingleMessage**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\AssertUtil.html#assertNullSingleMessage(java.lang.String, java.lang.Object))(java.lang.String partMessage, java.lang.Object value) |
| static void | [**assertTrue**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\AssertUtil.html#assertTrue(java.lang.String, boolean))(java.lang.String msg, boolean value) |

org.jdataex.common

**类 CommonUtil　常用判断封装**

|  |  |
| --- | --- |
| **方法** | |
| **限定符和类型** | **方法和说明** |
| static boolean | [**isBlank**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\CommonUtil.html#isBlank(java.lang.String))(java.lang.String content)  判断字符串是否为空（包括字符串对象为空）。 |
| static boolean | [**isNotBlank**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\CommonUtil.html#isNotBlank(java.lang.String))(java.lang.String content)  判断字符串是否为不空（包括字符串对象为不空）。 |
| static boolean | [**isNotNull**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\CommonUtil.html#isNotNull(java.lang.Object))(java.lang.Object obj)  判断实例是否为非空。 |
| static boolean | [**isNotNullValue**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\CommonUtil.html#isNotNullValue(java.lang.Object))(java.lang.Object obj) |
| static boolean | [**isNull**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\CommonUtil.html#isNull(java.lang.Object))(java.lang.Object obj)  判断实例是否为空。 |
| static boolean | [**isNullOrBlank**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\CommonUtil.html#isNullOrBlank(java.lang.Object))(java.lang.Object obj)  判断字符串是否为空（包括字符串对象为空）。 |
| static boolean | [**isNullValue**](file:///G:\code\j2se\workspace\JDataEx\api_javadoc\org\jdataex\common\CommonUtil.html#isNullValue(java.lang.Object))(java.lang.Object obj)  判断目标实例对象是否是象征为空值。 |