dgMaster

A simple, free, extensible, open source data generator

(An ongoing project)

Extensions made to the latest public release

July 2013

Contents

[New Concepts 3](#_Toc363114107)

[Stages 3](#_Toc363114108)

[MasterBuilder 3](#_Toc363114109)

[Configuration 3](#_Toc363114110)

[Changes and Improvements 6](#_Toc363114111)

[Application Context 6](#_Toc363114112)

[Randomizers 6](#_Toc363114113)

[GUI 6](#_Toc363114114)

## New Concepts

### Stages

We introduced a **stage** concept, which is a way of combining multiple output files into an aggregated iteration. With this mechanism in action it is possible to chain stages in order to achieve a pipeline on which a stage can output data into another. One stage is equivalent to a single standalone execution, which means that you can use the GUI to configure and execute it.

### MasterBuilder

**MasterBuilder** is the component that coordinates the stages creation and execution. This is where it is configured how many stages should be created, and how they should be executed. The MasterBuilder can be executed only in a standalone method, and for now no GUI support is delivered to control and configure this component.

### Configuration

In the project root folder you will find the DataBuild folder that contains the main structure on which the configuration and execution relies.

The following structure shows a typical configuration with one stage:

* **DataBuild/**
  + **conf/** [Masterbuilder configuration ]
    - application.properties
    - generator.xml
    - SystemDefinitions.xml
  + **logs/**  [logs folder]
  + **staging/** [Stages configuration and file handling]
    - **shared/** [intended for shared items between stages]
    - **stage1/**  [one folder per stage. Ex: stage1, stage2, stage3]
      * **conf/** [stage configuration]
        + stage.properties
        + master\_repository.xml
        + OutputFileDefinitions.xml
      * **dict/** [dictionary files to be used by this stage]
      * **input/** [input files from where the stage generators read]
      * **output/** [output files generated by the stage execution]
      * **stats/** [statistics files used by the stages generators]

#### Master builder configuration

##### application.properties

Below you can find a typical masterbuilder configuration file:

Note that you can change the name and location of:

* system definitions file
* The DataBuild folder
* log4j configuration file
* Staging folder
* Shared and Stages folders

# masterbuild home

# (all files and dirs are relative to this )

dgmaster.masterbuild.basedir=C:/workspace/DgMaster/DataBuild

#Log4j configuration file

dgmaster.masterbuild.log4j.config=generator.xml

#System definitions file - this is the file that contains the list of all available randomizers

dgmaster.masterbuild.default.systemdefinitions=SystemDefinitions.xml

# Total number of build stages

dgmaster.masterbuild.numstages=3

###################################################

# Order of build (defaults to time)

# time => for each day build stage1 to n

# stage => for each stage build day 1 to n

###################################################

dgmaster.masterbuild.buildorder=time

###################################################

# stage dirs and order of build

dgmaster.masterbuild.stagebasedir=staging

# stage dir shared by all stages

dgmaster.masterbuild.sharededir=staging/shared

###################################################

# Global properties

# ( can be overwritten in stagepropertiesfile)

# Start Date ddMMyyyy ( no ... you can't change the format )

dgmaster.masterbuild.start.date=01012013

dgmaster.masterbuild.number.of.days=10

###################################################

# Stage properties

###################################################

# Stage 1

###################################################

dgmaster.masterbuild.stagedir.1=stage1

dgmaster.masterbuild.stagebuilder.1=generator.stagebuild.impl.GenericStageBuilder

dgmaster.masterbuild.stagepropertiesfile.1=stage.properties

###################################################

#Stage 2

###################################################

dgmaster.masterbuild.stagedir.2=stage2

dgmaster.masterbuild.stagebuilder.2=generator.stagebuild.impl.EntityStageBuilder

dgmaster.masterbuild.stagepropertiesfile.2=stage.properties

###################################################

# Stage 3

###################################################

dgmaster.masterbuild.stagedir.3=stage3

dgmaster.masterbuild.stagebuilder.3=generator.stagebuild.impl.EventStageBuilder

dgmaster.masterbuild.stagepropertiesfile.3=stage.properties

#

## End of properties.

##### Stage configuration

###### Stage.properties

This file holds configuration specific to the stage.

###################################################

# dgmaster stage properties

###################################################

#Suffix of the files that hold the randomiser instances

#they will all be found and used in runtime

dgmaster.stagebuild.repository.suffix=repository.xml

#Suffix of the files that hold the outputfiles definition

#they will all be found and used in runtime

dgmaster.stagebuild.definition.suffix=definitions.xml

###### Repository configuration

These are the files where you save your randomizer instances configuration (ex: master\_repository.xml). This is an xml file, and it can also be generated and configured via GUI.

###### Output files configuration

These are the files where you configure the stage output files:

* Output location
  + Because the masterbuilder generates files for a given number of days (in one stage), the the output file name will be the one configured here, but modified to include the date for which the file was generated.
  + Example: If you configure that the file name is “test.txt”, and you execute the masterbuilder to generate data for 2 days starting on Jan 1st, you would find the following files in the output dir: *20130101-test.txt*, *20130102-test.txt* .
* Number of records
* File structure

This is an xml file, and it can also be generated and configured via GUI.

## Changes and Improvements

### Application Context

The *ApplicationContext* class had a few changes to make it more versatile and adaptable to different configurations scenarios:

* Base configuration Folder – This holds the folder path where you can find the output file definitions and the randomiser instances files. This value is changes in runtime, for every time a new stage is started.
* Definitions configuration folder - added this configuration which is the folder where you can find the log4j, systemdefinitions, dbDrivers and JavaDBMappings files.
* *refreshRandomiserInstances, refreshFileDefinitions, refreshRandomiserTypes-* these methods were changed in order to make them search for the configured file names, instead of always searching for a hardcoded filename. So is related to the stage.properties configuration. If these properties are not set, then the default names apply.

### Randomizers

A new interface - *IStageAwareRandomiserFunctionality* - was created to categorize randomisers that are stage aware. As an example, one randomiser was created with the objective of having an output file column where it shows the stage run date, i.e. the date for which the stage was generating data.

The example randomiser is implemented in generator.randomisers. StageRunDateSequencer .

**ReferentialListRandomiser**

Returns a value from a lookup list determined by the value of a previously created field .

Parameters :

referentialPosition

Position of the field to be used to match against the keyfield in the lookup list.

keyPosition

Position of the field in the lookup list that the referential field will be matched against.

targetPosition

Position of the field in the lookup list to return.

inputFile

The file containing the lookup list values.

separator

Field separator used in the input file.

EG :

We want to match a country code with a country name . We select the country code by some distribution so all we need to do is match it to the name to get a value.

So our fields are ...

CountryCode|CountryName

We have a lookup list of codes - > names such as ...

IE|Ireland

UK|United Kingdom

US|United states

So now the definition for CountryName will be .....

**referentialPosition**=0

**keyPosition**=0

**targetPosition**=1

**ReferentialPDFDiscreteRandomiser**

Returns a value from a lookup list determined by the value of a previously created field and the probability of the field.

Parameters :

referentialPosition

Position of the field to be used to match against the keyfield in the lookup list.

keyPosition

Position of the field in the lookup list that the referential field will be matched against.

targetPosition

Position of the field in the lookup list to return.

inputFile

The file containing the lookup list values.

countPosition

Position of the field in the lookup list that contains the count of occurrences .

EG: We have a field for account\_type and we need to create a field for transaction\_type based on it’s probability for that account type.

So our fields are ...

AccounType|TransactionType

The lookup list is as follows ...

2 credittransfer Deposit account

4 deposit Deposit account

29 cashdeposit Deposit account

1 chequedeposit Deposit account

2 adminfee Current account

3 credittransfer Current account

19 deposit Current account

So now the definition for TransactionType will be .....

**referentialPosition**=0

**keyPosition**=2

**targetPosition**=1

**countPosition**=0

NOTE: The sum of count is divided by the number of same values at keyPosition so that totals do not have to sum to 100.

**ReferentialStdDeviationRandomiser**

Calculate a numerical value based on a standard deviation and returns it’s string representation. Parameters :

RefPos(n)

Position of the field to be used to match against the keyfield in the lookup list.

KeyPos(n)

Position of the field in the lookup list that the referential field will be matched against.

rangesNum

The number of reference to key references . (n)

amavgPos

Position of the field in the lookup list that contains the average.

amstdPos

Position of the field in the lookup list that contains the standard deviation.

inputFile

The file containing the lookup list values.

EG: We have a field for account\_type and a field for transaction\_type and we want to calculate the possible value of the transaction.

So our fields are ...

AccounType|TransactionType|Amount

The lookup list is as follows ...

69.0764 9.164 credit transfer Deposit account

210.4486 16.17147888 deposit Deposit account

678.9 345.87 cash deposit Deposit account

348.8921739 87.89 cheque deposit Checking account

234.72 134.78 card debit Checking account

56 34.78 atm debit Checking account

626.92 85.128576 Cheque Payment Checking account

100 10 Branch Withdrawal Checking account

amavgPos =0

amstdPos=1

rangesNum =2

RefPos0=0

KeyPos0=3

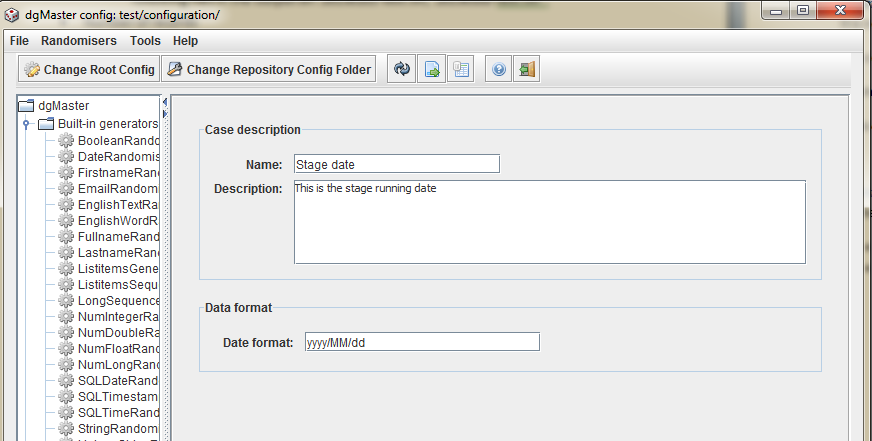
RefPos1=1

KeyPos1=2

### GUI

#### StageRunDateSequencer Panel

Along with the new randomiser, a panel was created for its configuration in the GUI. The panel is generator.panels. PanelStageRunDateRandomiser .



#### Main.java

It is now possible to start the GUI pointing to a specific configuration folder.

The command line parameters are: <Definitions folder> <Base configuration folder>

Example: Editting the stage1 configuration

*java.exe -jar dgMaster.jar DataBuild/conf/ DataBuild/conf/staging/stage1/conf/*

#### MainForm.java

In the GUI, two new buttons were added in order to change the configuration folders:

* **Change Root Config**: it allows you to search for a different definitions configuration folder.
* **Change Repository Config Folder**: it allows you to search for a different base configuration folder.

