org.jax.mgi.shr.dla.loader Design Document

Author: Mike Walker

Created: March 20, 2003

Last Modified: December 1, 2004 10:06

1 Purpose of Document

This document describes the classes belonging to the org.jax.mgi.shr.dla.loader package from the lib_java_dla product and provides source code examples for common usage patterns.

2 Introduction

The org.jax.mgi.shr.dla.loader package is comprised of classes that are responsible for providing basic functionality that can be shared across many loader implementations.

This package integrates with the lib_java_core product and the lib_java_dbsmgd and lib_java_dbsrdr products.

You must be pointing to Java1.4 in your classpath when using this product.

3 Overview of Classes

The **DLALoader** is an abstract class for performing data loads. It provides the 'basic-needs' for a general load application which include a DLALogger, and SQLStreams, SQLDataManagers and BCPManagers for both the target load database and the qc reporting database. There are four abstract classes which are implemented by the subclasses. They are initialize(), preprocess(), run(), and postprocess().

The **DLAStart** contains the main. It's purpose is to start loader applications from the command line. The load application is provided as a configuration parameter (either through a configuration file or though the java system properties).

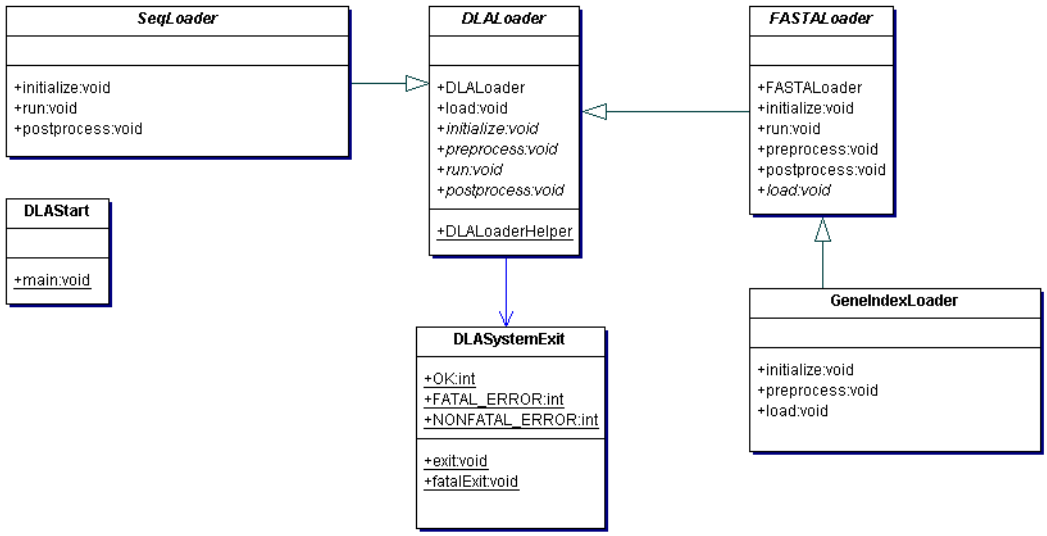
The **DLASystemExit** class is used to encapsulate the system exit details of the application which include conventions in logging and determination of the return status values provided back to the system command line.

The **FASTALoader** class is an additional loader base class which extends **DLALoader** and adds functionality that is unique to loading files in FASTA format. This functionality includes the parsing of FASTA data files. This class is abstract and provides the abstract method load(FASTAData) which is used to load one record of FASTA data into the database.

The **GeneIndexLoader** class is an extension of the **FASTALoader** class. It is responsible for loading gene index files that are in FASTA format. The NIA, TIGR and DoTS loads are all based on the **GeneIndexLoader** class.

The **SeqLoader** class is an extension of **DLALoader**. It provides base functionality for loading sequence data. The swiss-prot, tremble, genbank and assembly loads all are based on the **SeqLoader** class.

4 Class Diagram



5 Configuration

Configuration is accomplished by use of the org.jax.mgi.shr.config package from the lib_java_core product. See the documentation for this package for more information about the config base classes. There are a multitude of configurators which are used by classes within this package. Each configurator is responsible for configuring one of the loaders documented in section 3. There is the **DLALoaderCfg** used for configuring the **DLALoader** class, the **GeneIndexLoadCfg** used for configuring the **GeneIndexLoader** class, and the **SequenceLoadCfg** used for configuring the **SeqLoader** class. Because of protected access to methods within the config package that are required by the configurator classes, these classes must be a member of the org.jax.mgi.shr.config package. Details of how the configurator classes work can be found in the config documentation, but in brief, the configuration parameters are read in from configuration files and the java system properties. The configuration file names are indicated on the java command line by use of the CONFIG system property (see example 1).

EXAMPLE 1. setting the configuration

```
java -DCONFIG=configfile -DPARM1=OVERRIDE -DPARM2=OVERRIDE2 <app>
```

The configuration parameters are as follows:

• DLALoaderCfg Configuration Values

DLA_LOADER

The name of the loader class which is to be instantiated and run from the main. The **DLAStart** class uses this parameter. There is no default value for this parameter and the configurator will throw an exception if it's value cannot be found.

DLA_DB_PREFIX

The prefix used within the configuration file to distinguish the connection parameters for the load database. For eaxample, if your load database is MGD, then you would have connection parameters designated in the configuration file like MGD_DBURL, MGD_DBSERVER. The default is 'MGD'.

DLA_LOAD_STREAM

The name of the **SQLStream** class used for loading data into the database. The default is org.jax.mgi.shr.dbutils.dao.Inline_Stream.

DLA_QC_STREAM

The name of the **SQLStream** class used for loading data into the qc reporting database. The default is org.jax.mgi.shr.dbutils.dao.Inline_Stream.

DLA_TRUNCATE_LOAD_TABLES

A list of comma seperated names for load tables that you want the DLALoader class to automatically truncate at startup. There is no default. If no value is found, then truncation will not be performed on any load tables.

DLA_TRUNCATE_QC_TABLES

A list of comma seperated names for qc tables that you want the DLALoader class to automatically truncate at startup. There is no default. If no value is found, then truncation will not be performed on any qc tables.

• SequenceLoadCfg Configuration Values

JOBSTREAM

The name of the job stream. This value must be found in the MGI_User table or an exception will be raised. There is no default value. An exception will be raised if it's value cannot be found.

SEQ_LOAD_MODE

The mode of the sequence load, which can be either 'incremental' or 'delete_reload'. There is no default value. An exception will be raised if it's value cannot be found.

SEQ_VIRTUAL

This value is a boolean which designates whether or not the sequences being loaded are virtual (computationally derived). There is no default value. An exception will be raised if it's value cannot be found.

SEQ_MGITYPE

The mgi type of the data from this load. There is no default value. An exception will be raised if it's value cannot be found.

SEQ_LOGICALDB

The logical db type of the data from this load. The value must match some entry from the ACC_LogicalDB table or an exception will be raised. There is no default value. An exception will be raised if it's value cannot be found.

SEQ_PROVIDER

The name of the sequence provider. The value must match some entry from the VOC_Term table with vocabulary type of 'Sequence Provider'or an exception will be raised. There is no default value. An exception will be raised if it's value cannot be found.

SEQ_REPEAT_FILE

The name of the sequence repeat file where sequence numbers are placed if they are repeats. There is no default value. An exception will be raised if it's value cannot be found.

SEQ_QUALITY

The quality of the sequence (high, medium, low, etc). The value must be found in the VOC_Term table with a vocabulary type of 'Sequence Quality' or an exception will be raised. There is no default value. An exception will be raised if it's value cannot be found.

SEQ_TYPE

The type of sequence (RNA, DNA, etc). The value must be found in the VOC_Term table with a vocabulary type of ‘Sequence Type’ or an exception will be raised. There is no default value. An exception will be raised if it’s value cannot be found.

SEQ_ORGANISM

The organism represented by sequences of this load. The value must be found in the MGI_Organism table or an exception will be raised. There is no default value. An exception will be raised if it’s value cannot be found.

SEQ_RELEASE_NO

The release number for this load. There is no default value. An exception will be raised if it’s value cannot be found.

SEQ_STATUS

The status of the sequences for this load (active, deleted, split). The value must be found in the VOC_Term table with a vocabulary type of ‘Sequence Status’ or an exception will be raised. There is no default value. An exception will be raised if it’s value cannot be found.

SEQ_RELEASE_DATE

The release date of data for this load. There is no default value. An exception will be raised if it’s value cannot be found.

SEQ_JNUMBER

The bibliography reference (JNumber) for this load. There is no default value. An exception will be raised if it’s value cannot be found.

• CoordLoaderCfg Configuration Values

COORD_ORGANISM

Fill in the description here.

COORD_VERSION

Fill in the description here.

COORD_COLLECTION_NAME

Fill in the description here.

COORD_COLLECTION_ABBREV

Fill in the description here.

COORD_TYPE

Fill in the description here.

COORD_UNITS

Fill in the description here.

COORD_NAME

Fill in the description here.

COORD_ABBREV

Fill in the description here.

COORD_FEATURE_MGITYPE

Fill in the description here.

COORD_INTERPRETER

Fill in the description here.

COORD_PROCESSOR

Fill in the description here.

COORD_LOGICALDB

Fill in the description here.

COORD_REPEAT_FILE

Fill in the description here.

- **GeneIndexLoadCfg Configuration Values**

GNIDX_FILTER

The name of the class which implements the **FASTAFilter** interface. This class filters a **FASTAData** element by changing it or returning null (declines it). There is no default value. No filtering will be performed if this value is null.

•

6 DLALoader Class Details

The DLALoader class will responsible for initializing the system, logging standard messages to the log files, calling the callback methods on the subclass and performing standard system exiting.

The DLALoader class contains the following instance variables. In these descriptions, there is a distinction made between the 'load' database and the 'qc' database. The load database refers to the database where the data will be loaded and the qc database refers to the database where the qc data will be stored.

DLALogger logger - for sending messages to the four standard log files

SQLDataManager loadDBMgr - database manager for the load database

SQLDataManager qcDBMgr - database manager for the qc database

BCPManager loadBCPMgr - bcp manager for the load database

BCPManager qcBCPMgr - bcp manager for the qc database

SQLStream loadStream - SQLStream for loading into the load database

SQLStream qcStream - SQLStream for loading into the qc database

DLAExceptionHandler exceptionHandler - an exception handler

InputDataCfg inputConfig - the configurator for the input data file

DLALoaderCfg dlaConfig - the configurator for the DLALoader parameters.

This is an abstract class with the following abstract methods. The subclass will be responsible for implementing each of these methods.

protected abstract void initialize() throws MGIException

protected abstract void run() throws MGIException

protected abstract void preprocess() throws MGIException

protected abstract void postprocess() throws MGIException

The **initialize**() method is used to initialize any additional instance variables from the subclass. All base class instance variables are instantiated by the base class. If there are no additional instance variables, then this method can be implemented with an empty code block. All exceptions can be thrown if they extend **MGIException** (see the exception package from lib_java_core) and they will get handled by the **DLALoader** class.

The **preprocess**() method is used for any preprocessing required by the subclass. If there is none, then this method can be implemented with an empty code block. All exceptions can be thrown if they extend **MGIException** (see the exception package from lib_java_core) and they will get handled by the **DLALoader** class.

The **run**() method contains the actual loading process for this specific load. All exceptions can be thrown if they extend **MGIException** (see the exception package from lib_java_core) and they will get handled by the **DLALoader** class.

The **postprocess**() method is for closing any additional resources opened by the subclass. The base class will be responsible for closing all the instance variables it instantiated. If there are no additional resources created by the subclass then this method can be implemented with an empty code block. All exceptions can be thrown if they extend **MGIException** (see the exception package from lib_java_core) and they will get handled by the **DLALoader** class.

7 FASTALoader Class Details

This class extends **DLALoader**. It adds functionality which pertains to loading data from a FASTA formatted file. It implements the initialize(), preprocess(), run() and postprocess() methods from the **DLALoader** class. It defines one abstract class called load(FASTAData). This method is used for processing one FASTA record. The **GeneIndexLoader** extends this class and implements the load(FASTAData) method.

The additional instance variables for this class are as follows:

FASTAInputFile inputFile - the file containing the load data

SeqProcessor seqProcessor - the class which loads sequences into the db

8 GeneIndexLoader Class Details

This class extends **FASTALoader**. It adds functionality which pertains to loading gene index data. The loads that utilize this class are the NIA, TIGR, and DoTS loads. This class implements the load(FASTAData) method from the **FASTALoader** class. It is not abstract and can be considered as a complete gene index load. The particular differences between each of the flavors of gene index loads such as NIA or TIGR are all controlled by configuration parameters, most of which belong to the **SequenceLoadCfg** class.

The additional instance variables for this class are as follows:

FASTAFilter fastaFilter - a class for filtering incoming data
GeneIndexLoadCfg geneIndexCfg - the configurator for this class
SequenceLoadCfg seqcfg - the configurator for the sequence load parameters
MSRawAttributes msAttr - raw attributes for the sequence processor input
AccessionRawAttributes accAttr - raw attributes for the sequence processor input
SequenceRawAttributes seqAttr - raw attributes for the sequence processor input
RefAssocRawAttributes refAttr - raw attributes for the sequence processor input

9 SeqLoader Class Details

TBD

10 CoordLoader Class Details

TBD