RecordHandler

RecordHandler

A tool to help organize and unify data transactions while outside of an RDF model

This tool is used for the storage of data before it is put into an RDF model. For the initial fetch and post translation the information is stored in this type of structure. The default behavior is to list the record ids contained in the Record Set given to it as input. Optionally, you can inspect/modify a records contents.

RecordHandler Parameters

wordiness - (optional) sets the lowest level of log messages to be displayed to the console. The lower the log level, the more detailed the messages.

Possible Values:

- <Param name="wordiness">OFF</Param> - (Default) Results in no messages being displayed other than requested record listing/contents printed to stdout.
- <Param name="wordiness">ERROR</Param> - Results in only messages from the ERROR level to be displayed. Error messages detail when the tool has experienced an error preventing it from completing its task.
- <Param name="wordiness">WARN</Param> - Results in only messages above and including WARN level messages to be displayed. RecordHandler does not produce any WARN level messages.
- <Param name="wordiness">INFO</Param> - Results in all messages above and including INFO level messages to be displayed. INFO level messages detail when the tool has started and ended.
- <Param name="wordiness">DEBUG</Param> - Results in all messages above and including DEBUG level messages to be displayed. DEBUG level will display stacktrace information if an error occurs.
- <Param name="wordiness">ALL</Param> or TRACE</Param> - Results in all messages above and including TRACE level messages to be displayed, since trace is the lowest level it is the same as ALL in practice. TRACE level messages are not produced by the RecordHandler tool.

recordId - (optional) specifies the recordId to inspect/modify. If no value parameter is given, the recordscontents will be printed

Example:

- <Param name="recordId">person5172951</Param> - selects the record identified by the id 'person5172951'
- <Param name="recordId">89187aca01</Param> - selects the record identified by the id '89187aca01'
- <Param name="recordId">http://local.edu/harvest/people/n1512</Param> - selects the record identified by the id 'http://local.edu/harvest/people/n1512'

value - (optional - only valid when recordId is provided as well) specifies the new value for the record that is identified by the given recordId

Example:

- <Param name="value">blah, this is my new content</Param> - sets the new content of the record identified by the given recordId to be 'blah, this is my new content'

output-file - (optional - not used when value is set) specifies a file to be the output destination for record listing/contents

Example

- <Param name="output-file">/absolute/path/to/file</Param> - An absolute path to a file on linux/unix/macosx operating systems
- <Param name="output-file">C:/absolute/path/to/file</Param> - An absolute path to a file on a windows operating system
- <Param name="output-file">relative/path/to/file</Param> - A path to a file that is relative to the folder the shell was in when this command was executed

input-config - (optional - at least one of this and/or inputOverride) the configuration file that describes the input record set. The parameters for this config file are described in the Record Sets section below.

Example:

- <Param name="input-config">/absolute/path/to/file.conf.xml</Param> - An absolute path to a recordhandler config file on linux/unix/macosx based systems.
- <Param name="input-config">C:/absolute/path/to/file.conf.xml</Param> - An absolute path to a recordhandler config file on a windows operating system.
- <Param name="input-config">relative/path/to/file.conf.xml</Param> - A path to a recordhandler config file that is relative to the folder the shell was in when this command was executed.

inputOverride - (optional - at least one of this and/or input) specify the parameters for the record set without a config file and/or override specific parameters from the given config file. The parameters that can be set/overridden are described in the Record Sets section below.

Example:

- <Param name="inputOverride">paramName=valueToUse</Param>

Record Sets

rhClass - The class for the handler for this record set

Example Values:

- <Param name="inputOverride">rhClass=org.vivoweb.harvester.util.repo.TextFileRecordHandler</Param> - to store each record as a file in a folder (specified by fileDir). NOTE: performance with TextFileRecordHandler is quite poor, but is the suggested while building and testing your script. Once the script works, switch to using a higher performance recordhandler like JDBCRecordHandler (in an h2 database) or JenaRecordHandler (in a tdb model).
- <Param name="inputOverride">rhClass=org.vivoweb.harvester.util.repo.JDBCRecordHandler</Param> - to store each record in a table in a relational database (specified with dbClass, dbUrl, dbUser, dbPass, dbTable, and dataFieldName)
- <Param name="inputOverride">rhClass=org.vivoweb.harvester.util.repo.JenaRecordHandler</Param> - to store each record in a jena triple store (specified with dataFieldType, jenaConfig, and/or all the parameters for a jena model (see below)

TextFileRecordHandler Parameters

fileDir - the directory in which to store the files for each record

Example Values:

- <Param name="inputOverride">fileDir=/absolute/path/to/dir</Param> - An absolute path to a directory on linux/unix/macosx operating systems.
- <Param name="inputOverride">fileDir=C:/absolute/path/to/dir</Param> - An absolute path to a directory on a windows operating system.
- <Param name="inputOverride">fileDir=relative/path/to/dir</Param> - A path to a directory that is relative to the folder the shell was in when this command was executed.

JDBCRecordHandler Parameters

dbClass - the JDBC driver class to use

Example Values:

- <Param name="inputOverride">dbClass=com.mysql.jdbc.Driver</Param> - for a mysql database
- <Param name="inputOverride">dbClass=org.h2.Driver</Param> - for an h2 database

dbUrl - the JDBC connection url

Example Values:

- <Param name="inputOverride">dbUrl=jdbc:mysql://127.0.0.1:3306/dbName</Param> - for a mysql database.
  - See http://dev.mysql.com/doc/refman/5.6/en/connector-j-reference-configuration-properties.html
- <Param name="inputOverride">dbUrl=jdbc:h2:path/to/h2/store</Param> - for an h2 database.
  - See http://www.h2database.com/html/features.html#database_url

dbUser - the DB username to use

Example Values:

- <Param name="inputOverride">dbUser=sa</Param> - used for h2 database (the default h2 system admin login)
- <Param name="inputOverride">dbUser=myUser</Param>

dbPass - the DB password to use

Example Values:

- <Param name="inputOverride">dbPass=</Param> - used for h2 database (the default h2 system admin login)
- <Param name="inputOverride">dbPass=myPass</Param>

dbTable - (optional) the name of the table to store data in, if non-existent will be created

Example Values:

- (default) <Param name="inputOverride">dbTable=recordTable</Param>
- <Param name="inputOverride">dbTable=myTableName</Param>

dataFieldName - (optional) the name of the field to use, if table is non-existent will be created with table

Example Values:

- (default) <Param name="inputOverride">dataFieldName=dataField</Param>
- <Param name="inputOverride">dataFieldName=myDataFieldName</Param>

JenaRecordHandler Parameters

dataFieldType - (optional) the predicate to use for storing data properties

Example Values

- (default) <Param name="inputOverride">dataFieldType=http://vivoweb.org/harvester/rh#dataFieldType</Param>
- <Param name="inputOverride">dataFieldType=http://yourmom.com/propbase#myProp</Param>

jenaConfig - (optional - at least one of this and/or full set of params defining a model as described in Models section below) the configuration file that describes the model in which to store data. The parameters for this config file are described in the Models section below.

Example Values:

- <Param name="inputOverride">jenaConfig=/absolute/path/to/jena-model.config.xml</Param> - An absolute path to a directory on linux/unix /macosx operating systems
- <Param name="inputOverride">jenaConfig=C:/absolute/path/to/jena-model.config.xml</Param> - An absolute path to a directory on a windows operating system
- <Param name="inputOverride">jenaConfig=relative/path/to/jena-model.config.xml</Param> - A path to a directory that is relative to the folder the shell was in when this command was executed

Recordhandler programmatic view

RecordHandler handles both input and output of records from data repositories. By abstracting CRUD operations for the different data repositories, the system can interact with records in a standardized way.

The variety of record handlers allows flexibility and customization while using the VIVO Harvester. The most frequently used recordhandlers are the TextFileRecordHandler and the JDBCRecordHandler.

Record Class

```
public class Record
```

Constructors

- public constructor(recID:String, recData:String)

Mutators

- public setData(newData:String, operator:Class<?>) : void
- public setProcessed(operator:Class<?>) : void

Accessors

- public getID() : String
- public getData() : String

Record Meta Data Class

```
public class RecordMetaData implements Comparable<RecordMetaData>
```

Contains a transaction log entry for a write or process operation

Constructors

- protected constructor(operationDate:Calendar, operatorClass:Class<?>, operationType:RecordMetaDataType, md5:String)
- protected constructor(operatorClass:Class<?>, operationType:RecordMetaDataType, md5:String)
    - simply creates cal:Calendar (based on now in GMT, with USA locale) and calls constructor(cal, operatorClass, operationType, md5)

Accessors

- public getDate() : Calendar
- public getOperator() : Class<?>
- public getOperation() : RecordMetaDataType
- public getMD5() : String

Utilities

- public static enum RecordMetaDataType
    - written
    - processed
    - error
- public static makeMD5Hash(text:String) : String
- public compareTo(RecordMetaData o) : int
    - for Comparable<RecordMetaData>
    - will allow sorting in reverse-chronological order

Record Handler Abstract Class

```
public abstract class RecordHandler implements Iterable<Record>
```

By extending Iterable, you will be able to do fun stuff like:

```
RecordHandler recSet = new ExampleRecordHandler();
for(Record rec : recSet){
  log.trace("================================================================");
  log.trace("Record "+rec.getID()+":");
  log.trace("----------------------------------------------------------------");
  log.trace(rec.getData());
}
```

Mutators

- public abstract addRecord(rec:Record, creator:Class<?>, overwrite:boolean) : void
- public addRecord(recID:String, recData:String, creator:Class<?>, overwrite:boolean) : void
    - simply creates rec:Record using the params and calls addRecord(rec, creator, overwrite)
- public addRecord(rec:Record, creator:Class<?>) : void
    - simply calls addRecord(rec, creator, isOverwriteDefault())
- public addRecord(recID:String, recData:String, creator:Class<?>) : void
    - simply creates rec:Record using the params and calls addRecord(rec, creator, isOverwriteDefault())
- protected abstract addMetaData(rec:Record, rmd:RecordMetaData)
- protected addMetaData(rec:Record, operator:Class<?>, type:RecordMetaDataType) : void
    - simply calls addMetaData(rec, new RecordMetaData(operator, type, RecordMetaData.makeMD5Hash(rec.getData())))
- protected setProcessed(rec:Record, operator:Class<?>) : void
    - simply calls addMetaData(rec, operator, RecordMetaDataType.processed)
- protected setWritten(rec:Record, operator:Class<?>) : void
    - simply calls addMetaData(rec, operator, RecordMetaDataType.written)
- protected abstract delMetaData(recID:String) : void
- public delRecord(recID:String) : void
- public abstract setParams(params:Map<String,String>) : void
- public setOverwriteDefault(overwrite:boolean) : void

Accessors

- public abstract getRecordData(recID:String) : String
- public getRecord(recID:String) : Record
    - simply returns a Record created using recID and getRecordData(recID)
- public abstract getRecordMetaData(recID:String) : SortedSet<RecordMetaData>
- protected getLastMetaData(recID:String, type:RecordMetaDataType) : RecordMetaData
    - iterates through results of getRecordMetaData(recID) in descending order till it finds and returns a metadata of matching type
- public getLastMetaData(recID:String) : RecordMetaData
    - simply returns getRecordMetaData(recID, null)
- public getLastWrittenMetaData(recID:String) : RecordMetaData
    - simply returns getRecordMetaData(recID, RecordMetaDataType.written)
- public getLastProcessedMetaData(recID:String) : RecordMetaData
    - simply returns getRecordMetaData(recID, RecordMetaDataType.processd)
- public isOverwriteDefault() : boolean

Utilities

- public static parseConfig(filename:String) : RecordHandler

Map Record Handler

Stores records using Map<String,String> and metadata using Map<String,SortedSet<RecordMetaData>>

Constructors

- public constructor()

Mutators

- public addRecord(rec:Record, creator:Class<?>, overwrite:boolean) : void
- public delRecord(recID:String) : void
- public setParams(params:Map<String,String>) : void
    - unused as no params exist
- protected addMetaData(rec:Record, rmd:RecordMetaData) : void
- protected delMetaData(recID:String) : void

Accessors

- public getRecordData(recID:String) : String
- public getRecordMetaData(recID:String) : SortedSet<RecordMetaData>

Utilities

- public iterator() : Iterator<Record>

Text File Record Handler

`public class` **`TextFileRecordHandler`** `implements` *`RecordHandler`*

Stores each record in location fileDir (resolved using apache commons vfs) with filename recID filled with recData. Allows for access to files located in (S) FTP, Local File System, HTTP(S), Temporary Files, Archive Files (Zip, Jar, Tartgz/tbz2, gzip, bzip2), res, ram, mime. Meta data for each record is stored in subdirectory fileDir/.metadata in individual files.

Constructors

- protected constructor()
    - called by RecordHandler config parser... actually initialized by setParams(params)
- public constructor(fileDir:String)

```
new TextFileRecordHandler("XMLVault");
new TextFileRecordHandler("ftp://username:password@127.0.0.1:21/path/to/dir");
```

Mutators

- public addRecord(rec:Record, operator:Class<?>, overwrite:boolean) : void
- public delRecord(recID:String) : void
- public setParams(params:Map<String,String>) : void
- protected addMetaData(rec:Record, rmd:RecordMetaData) : void
- protected delMetaData(recID:String) : void

Accessors

- public getRecordData(recID:String) : string
- public getRecordMetaData(recID:String) : SortedSet<RecordMetaData>

Utilities

- public iterator() : Iterator<Record>

JDBC Record Handler

`class` **`JDBCRecordHandler`** `implements` *`RecordHandler`*

Records are stored in a single, 3 column table:

<tableName>		
UNIQUE_AUTOGENERATED_ID	**recordID**	**<dataFieldName>**
primary_key:int(10)	unique_key:varchar (100)	longtext
<auto_generated_id>	<rec.getID()>	<rec.getData()>

Record meta data is stored in a separate 6 column table:

<tableName>_rmd					
UNIQUE_AUTOGENERATED_RMD_ID	**record_autogen_id**	**utc_milli_from_epoch**	**operation**	**operator**	**md5**
primary_key:int(10)	foreign_key(tableName):int(10)	index:int(25)	varchar(10)	varchar(100)	int(32)
<auto_generated_id>	<UNIQUE_AUTOGENERATED_ID>	<rmd.getDate().getTimeInMillis()>	<rmd.getOperation()>	<rmd.getOperator().getName()>	<rmd.getMD5()>

JDBCRecordHandler connects to a database located at connLine using the jdbcDriverClass, username, and password. Performs the CRUD operations using the tableName, idFieldName, and dataFieldName.

Constructors

- protected constructor()
    - called by RecordHandler config parser... actually initialized by setParams(params)
- public constructor(jdbcDriverClass:String, connLine:String, username:String, password:String, tableName:String, dataFieldName:String)
  connLine strings have the following format: "jdbc:%connType%://%host%:%port%/%dbName%"

```
new JDBCRecordHandler("com.mysql.jdbc.Driver", "jdbc:mysql://127.0.0.1:3306/dbName", "username", "password", "tableName", "dataFieldName");
```

- public constructor(jdbcDriverClass:String, connType:String, host:String, port:String, dbName:String, username:String, password:String, tableName:String, dataFieldName:String)
  builds a connLine from the connType, host, port, and dbName and calls the constructor that takes a connLine

  ```
  new JDBCRecordHandler("com.mysql.jdbc.Driver", "mysql", "127.0.0.1", "3306", "dbName", "username", "password", "tableName", "dataFieldName");
  ```

Mutators

- public addRecord(rec:Record, operator:Class<?>, overwrite:boolean) : void
- public delRecord(recID:String) : void
- public setParams(params:Map<String,String>) : void
- protected addMetaData(rec:Record, rmd:RecordMetaData) : void
- protected delMetaData(recID:String) : void

Accessors

- public getRecordData(recID:String) : string
- public getRecordMetaData(recID:String) : SortedSet<RecordMetaData>

Utilities

- protected finalize() : void
  - called upon desctruction... closes db connection
- public iterator() : Iterator<Record>

Jena Record Handler

`public class` **`JenaRecordHandler`** `implements` *`RecordHandler`*

Each record is stored as 3 triples:

Subject	Predicate	Object
<UNIQUE_AUTOGENERATED_URI>	rdf:type	ingestNS: record
<UNIQUE_AUTOGENERATED_URI>	ingestNS: idField	<recID>
<UNIQUE_AUTOGENERATED_URI>	<dataFieldType>	<recData>

JenaRecordHandler connects to a jena model named modelName located at connLine using the jdbcDriverClass, username, and password.

Constructors

- protected constructor()
  - called by RecordHandler config parser... actually initialized by setParams(params)
- public constructor(jdbcDriverClass:String, connType:String, host:String, port:String, dbName:String, username:String, password:String, dbType:String, modelName:String, dataFieldType:String)

  ```
  new JenaRecordHandler("com.mysql.jdbc.Driver", "mysql", "127.0.0.1", "3306", "dbName", "username", "password", "MySQL", "modelName", "http://localhost/demo#data");
  ```

- public constructor(jdbcDriverClass:String, connType:String, host:String, port:String, dbName:String, username:String, password:String, dbType:String, dataFieldType:String)

  ```
  new JenaRecordHandler("com.mysql.jdbc.Driver", "mysql", "127.0.0.1", "3306", "dbName", "username", "password", "MySQL", "http://localhost/demo#data");
  ```

- public constructor(configFile:String, dataFieldType:String)

  ```
  new JenaRecordHandler("JenaModelConfigFile.xml", "http://localhost/demo#data");
  ```

Mutators

- public addRecord(rec:Record, operator:Class<?>, overwrite:boolean) : void
- public delRecord(recID:String) : void
- public setParams(params:Map<String,String>) : void
- protected addMetaData(rec:Record, rmd:RecordMetaData) : void
- protected delMetaData(recID:String) : void

Accessors

- public getRecordData(recID:String) : string
- public getRecordMetaData(recID:String) : SortedSet<RecordMetaData>

Utilities

- public iterator() : Iterator<Record>