

DataForge

Multi Datasource Aggregation Engine

Ben Walding

DataForge: Multi Datasource Aggregation Engine

by Ben Walding

Published 2004

Copyright © 2004, 2003, 2002 Ben Walding

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Table of Contents

Example Glossary	7
1. Introduction	1
Overview	1
Prerequisites	1
2. Architecture	2
Overview	2
Features	2
Core	2
Caching	2
Design Decisions	3
Separate Configuration Tree / Execution Tree	3
Single output	3
3. Getting Started	4
Downloading	4
Configuration	4
A.df - DataForge Configuration file	4
A.java - Java Code	4
A.tsv - Tab delimited data file	5
A.properties - Data schema properties	5
4. Elements	6
Transformers	6
Annotations	6
Forges	6
Binds	6
Properties	6
5. Expressions	7
OGNL Expressions	7
SQL Expressions	7
6. Editor	8
Overview	8
7. Context	9
Overview	9
Properties	9
8. Debugger	10
Overview	10
How it works	10
9. Logging	11
Overview	11
A. GNU Free Documentation License	12
PREAMBLE	12
APPLICABILITY AND DEFINITIONS	12
VERBATIM COPYING	13
COPYING IN QUANTITY	13
MODIFICATIONS	14
COMBINING DOCUMENTS	15
COLLECTIONS OF DOCUMENTS	15
AGGREGATION WITH INDEPENDENT WORKS	16
TRANSLATION	16
TERMINATION	16
FUTURE REVISIONS OF THIS LICENSE	16
ADDENDUM: How to use this License for your documents	16
B. Transformers	18
AbstractTransformer	18

AppendTransformer	18
BeanShellTransformer	18
DebugTransformer	19
DeltaTransformer	19
DifferenceTransformer	19
DiskCacheTransformer	19
EmptyTransformer	20
FilterTransformer	20
ForEachTransformer	20
FormatTransformer	21
GroupByTransformer	21
HTMLPresentation	21
JoinTransformer	21
MemoryCacheTransformer	21
MultiplexTransformer	22
NullTransformer	22
ObjectPresentation	22
ResultSetJDBCTransformer	22
StaticTransformer	23
TSVTransformer	23
ThrottleTransformer	23
TopNTransformer	23
Transformer	24
XMLPresentation	24
XMLTransformer	24
Index	25

List of Figures

6.1. DataForge Configuration Editor 8

Example Glossary

D

DataForge Configuration Editor

The configuration editor used to edit DataForge configurations.

Chapter 1. Introduction

Overview

DataForge is a scalable engine for managing, running, caching and aggregating queries. Where most databases perform star style queries relatively quickly, they tend to fall apart when the joins become more complicated as their optimizers just can't handle multiple convoluted joins involving sub-selects etc.

DataForge makes it simple to transparently cache query results to disk or memory and then retrieve that data in a controlled manner. It is trivial to configure a cache such that snapshots of data are taken at daily intervals and then compared to produce further results.

The underlying goal of DataForge is to make writing flexible data access code as simple as possible. It is possible to introduce DataForge into your application quickly and simply as there is no huge framework to be part of, this crosses over into the testing arena where you can switch data environments with the flick of a switch.

DataForge supports native Java code inside queries using two different expression parsers.

Prerequisites

This reference assumes you have a strong grasp of the Java programming language. This reference also assumes an understanding of JDBC and XML.

Chapter 2. Architecture

Overview

Features

New features:

1. Use of non database data sources (eg directories)
2. XML config rather than database config?
3. XML config OR database config
4. Use of SQL as a modelling language?
5. SQL inheritance model

Core

The core concept within the land of DataForge is the transformer. So named because it performs transforms on data that passes through it.

Data flows into a transformer from multiple data sources and out to a single data sink. A single transformer can not feed to multiple data sinks as that would break the data "pull" model that is intrinsic to the model that DataForge follows (This may not be true indefinitely, by marking the stream of events and caching them, multiple readers could attach to a single stream - this would be a function of the transformer and not of the dataforge).

Generator	A generator is a transformer that produces data, but does not consume any from within the forge
Transformer	A transformer has multiple (1 - many) inputs and many outputs (1 - many, but usually only 1).
Presentation	A presentation transformer consumes data events, and produces a specific output - eg HTML, Excel (HSSF) or XML - Whatever you can come up with.

Caching

One of the reasons you may be looking at using DataForge is to improve the performance of queries / joins.

Using the `DiskCacheTransformer` / `MemoryCacheTransformer` allows you to bolt caching on at any level(s) of the forging process.

Consider that you have 2 queries that you want to join together. One is from the call logging system that lists all calls logged during the day. The other query is from the LDAP phone server that is running from a country far far away. By adding a `DiskCacheTransformer` in front of the LDAP transformer, the performance can be improved 10-fold.

The key difference between the Disk and Memory caches is that the Disk cache persists across JVM restarts whereas the Memory cache only lasts for as long as the DataForge. Of course it is quite reasonable to bolt a memory cache in front of the disk cache and get the best of both worlds!

Design Decisions

Separate Configuration Tree / Execution Tree

The two trees are separated for possible performance improvements and simplicity in programming. By separating them, configuration information is isolated to the configuration tree and doesn't cover up or place design constraints on the execution tree.

Single output

After careful consideration it has been decided that having a transformer able to feed data to multiple transformers simultaneously would be a poor choice. However multiple transformers can source a single transformer, in this case, the source transformer will be opened multiple times.

Chapter 3. Getting Started

Downloading

<http://dist.codehaus.org/dataforge/>

1. antlr (2.7.2) - <http://www.ibiblio.org/maven/antlr/jars/antlr-2.7.2.jar>
2. bsh (1.2b7) - <http://www.ibiblio.org/maven/bsh/jars/bsh-1.2b7.jar>
3. com.walding (SNAPSHOT) - <http://www.ibiblio.org/maven/com.walding/jars/com.walding-SNAPSHOT.jar>
4. log4j (SNAPSHOT) - <http://jakarta.apache.org/log4j/> <http://www.ibiblio.org/maven/log4j/jars/log4j-1.2.8.jar>
5. commons-logging (1.0.3) - <http://jakarta.apache.org/commons/logging/> <http://www.ibiblio.org/maven/commons-logging/jars/commons-logging-1.0.3.jar>
6. ognl (2.5.1) - <http://www.ognl.org/> <http://www.ibiblio.org/maven/ognl/jars/ognl-2.5.1.jar>
7. xpp3 (1.1.3.4-RC8) - <http://www.ognl.org/> <http://www.ibiblio.org/maven/xpp3/jars/xpp3-1.1.3.4-RC8.jar>

Configuration

For this example, create all the required elements in the same package. The java code assumes a package name of `example1`

A.df - DataForge Configuration file

Create the following as `A.df` in the same package

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE dataforge
  PUBLIC "-//DataForge//DTD DataForge 1.0//EN"
  "http://dataforge.codehaus.org/dataforge_1_0.dtd">
<dataforge>
  <title>Simple example of how to use a TSVTransformer</title>
  <transformer
    name="A"
    type="org.codehaus.dataforge.engine.transformers.TSVTransformer">
    <property name="source.data" value="A.tsv"/>
    <property name="source.schema" value="A.properties"/>
  </transformer>
</dataforge>
```

A.java - Java Code

Place the following class (A) in a `example1` package.

```
package example1;
import java.sql.ResultSet;

import org.codehaus.dataforge.engine.DataForge;
import org.codehaus.dataforge.engine.configuration.XMLConfiguration;
import org.codehaus.dataforge.engine.transformers.Transformer;
```

```
public class A {
    public static void main(String args[]) throws Exception {
        DataForge df = XMLConfiguration.quickConfig(A.class, "A.df");
        Transformer t = df.resolveTransformer("A");

        ResultSet r = t.open();
        System.out.println("Starting.");
        while (r.next()) {
            //Do something with this resultset
            System.out.println("New Row!");
            for (int i = 1; i <= r.getMetaData().getColumnCount(); i++) {
                System.out.println("Col[" + i + "] = "
                    + r.getMetaData().getColumnName(i) + " = "
                    + r.getObject(i));
            }
        }
        System.out.println("Done.");
        r.close();
    }
}
```

A.tsv - Tab delimited data file

Create the following as A.tsv in the same package

```
Plant Coordinates Name
37 3700101001 NORAD
37 3705500001 Northern Lights
60 6000100001 Space Command - Ural Mountains
60 6000100002 Spinnaker Windmaster
99 9905000000 Fidel Castro
```

A.properties - Data schema properties

Create the following as A.properties in the same package

```
##The header lines are ignored (for now. In future they may be treated as schema informati
header.count=1

0.column.name=Plant
0.column.type=com.walding.common.types.StringDataType
1.column.name=Coordinates
1.column.type=com.walding.common.types.StringDataType
2.column.name=Name
2.column.type=com.walding.common.types.StringDataType
```

Chapter 4. Elements

Transformers

transformer (see transformers)

The transformer is the basic building block of DataForge

Annotations

The annotation is simply a textual element that is typically used for documentation.

Forges

The forge element allows a query to build on standard building blocks that are provided by the system expert. With a forge included in your own forge, building a join between two disparate systems is as simple as wiring together a transformer with a transformer source prefixed by the namespace of that transformer.

Binds

The Bind element groups together the links between a transformers. Typically a bind has a single target (repeated in each Link element)

Properties

One major piece of brokenness is magic properties. i.e. properties which you have to know and are only documented in the source code. Unfortunately since we used DTD validation this will continue, but the documentation for each of the transformers should be improved shortly.

Chapter 5. Expressions

OGNL Expressions

OGNL stands for Object-Graph Navigation Language; it is an expression language for getting and setting properties of Java objects. You can find the complete OGNL reference at <http://ognl.org/>

The OGNL expression type is used for most other things that require some level of integration with strings. eg. "cache/\${year}"

Saying that DataForge uses an OGNL expression language only tells half the story. The expression language is actually a combination of OGNL and a small wrapper to allow simple embedding of OGNL expressions into strings.

For examples, given `a = "DataForge"`, `b = "somewhat"` and an expression of `"${a} is ${b.toUpperCase()} useful"`. Then typically within DataForge, this will evaluate to "DataForge is SOME-WHAT useful".

Only certain fields currently interpret expressions as OGNL, as time goes by, more of the system will become OGNL aware - it is not being used wholesale as it may clog the design of DataForge with something that has limited utility.

SQL Expressions

The SQL expression type is typically used in the transformers for highspeed operations that map closely to SQL types and functions.

Chapter 6. Editor

Overview

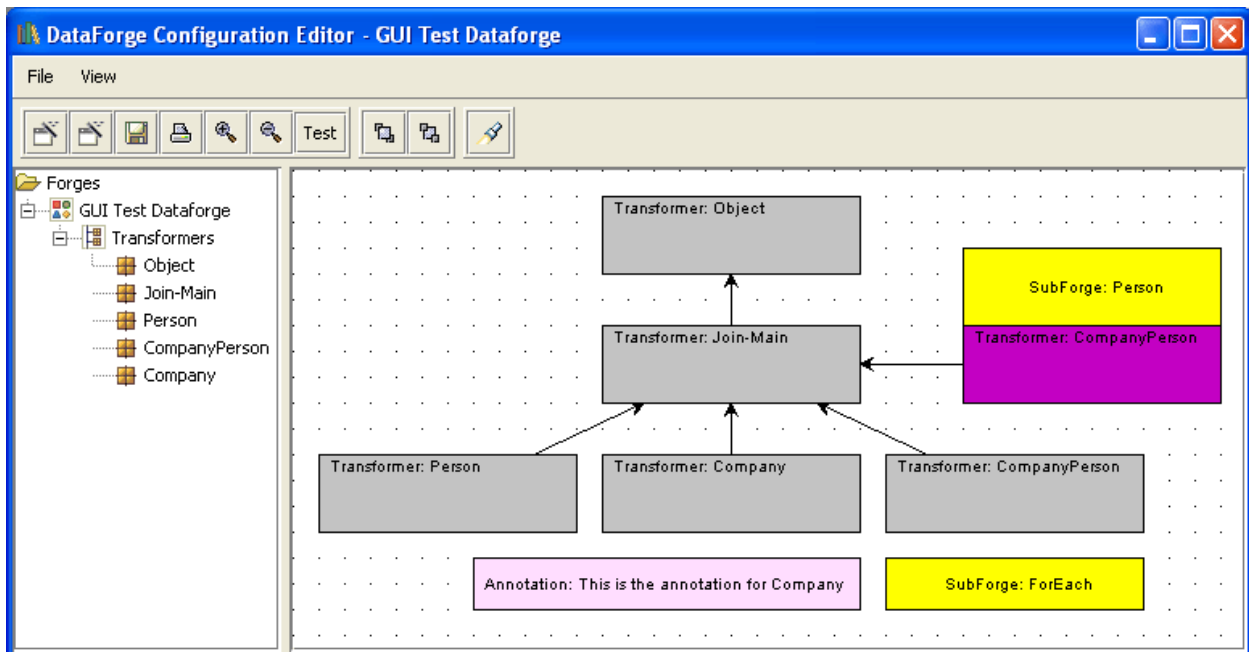
The DataForge Configuration Editor (DCE) is a GUI editor that aids in the configuration of DataForges. Irrespective of what any one tells you, it is not trivial to correctly configure the DataForge with the correct settings (using XML). This gets more difficult as the number of sub forges / additional transformers increases.

The DCE allows you to visualise the flow of information within the forge and view and edit multiple interrelated forges. With the recent addition of annotations, documenting your queries becomes less tedious.

The DCE also allows you to export graphical representations of the Forges suitable for documentation purposes.

In addition to allowing you to view existing configurations, you can also create new configurations using an unintuitive interface.

Figure 6.1. DataForge Configuration Editor



Chapter 7. Context

Overview

The purpose of the context is to consistently pass configuration through the system. By using a context, the DataForge is made resilient to thread safety issues such as configurations from separate threads overriding one another.

The examples shown throughout the system show the correct way to configure DataForges in a simple manner, however if one wants to push the boundaries of what can be done, then reading this chapter is advised.

Properties

There are two attributes for a property; name (String), value (Object). A property can not have a `null` name. A property can have a `null` value.

Due to how transformers are opened, each transformer must fully construct the context of the child transformers it is opening. This inversion of control is useful as it means that a transformer does not need to reach out to get properties.

Chapter 8. Debugger

Overview

This is a completely theoretical chapter. No debugger exists at present, but this is how I envisage and plan to implement it.

One major advantage of using DataForge is its ability to be debugged at run-time. Using simple configuration settings, the entire forge can be reconfigured such that it exposes itself to an external debugger.

The major advantage of doing this over just using the debugger architecture built into the JVM is that we can display the debug information in a DataForge aware manner.

How it works

Once you have given the command to debug the DataForge, all transformers become wrapped by the DebugTransformer. All ResultSet become wrapped by DebugResultSet. This allows DataForge to capture runtime information from the result sets and pass it to the debugger.

One of the earlier implementations of debugging used a set of hooks that a transformer could take advantage of to report of debug progress. This severely cluttered the transformer code and limited performance as it had to perform debug checks during normal runs.

Once the forge has been altered to support debugging (i.e. transformers and resultsets are wrapped inside debugger aware version), the debug manager will then expose itself via RMI to a DataForge debugger.

The debugger will be built into the editor such it displays a visual representation of what the forges are doing.

Chapter 9. Logging

Overview

All logging within DataForge is done using `log4j`, as such logging may be controlled in any manner that `log4j` normally allows.

It is recommended that you configure `log4j`'s default threshold in the `org.codehaus.dataforge` logger to the `WARN` level.

`FATAL` - fatal errors - Shut it all down and start again.

`ERROR` - recoverable errors

`WARN` - warning statements - Bad things have happened, but we'll continue on.

`INFO` - informational statements

`DEBUG` - debug statements

Appendix A. GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a

machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later ver-

sion published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Appendix B. Transformers

AbstractTransformer

Base implementation for most built-in DataForge transformers.

Provides the core infrastructure for creating a transformer.

Subclasses should endeavour not to override void open(PropertySet) as the AbstractTransformer provides a lot of functionality to do with property passthrough and control. Specifically it merges properties together, removes transformer specific properties, then delegates into the void openInternal(PropertySet)

author Ben Walding

version \$Id: AbstractTransformer.java,v 1.14 2004/03/11 00:20:31 bwalding Exp \$

AppendTransformer

Appends the results of opening all it's transformers together.

author Ben Walding

version \$Id: AppendTransformer.java,v 1.7 2004/03/03 16:19:58 bwalding Exp \$

BeanShellTransformer

The BeanShell code that you write should have the following function prototypes

```
RowEvent getNext();
```

```
TransformerResultSet open();
```

```
void close();
```

You can assume the following items will be available in your execution context: rs = the TransformerResultSet returned by the BeanShell "open" (available for getNext, close) owner = BeanShellTransformer (that owns this script) (available in all routines)

The lifecycle of the BeanShellTransformer is thus:

The transformer opens and parses the supplied bean shell file.

The interpreter verifies that the functions listed above are available.

The BST slots this into the owner variable.

The BST calls open and slots the result into the rs variable

author Ben Walding
version \$Id: BeanShellTransformer.java,v 1.10 2004/03/03 16:19:58 bwalding Exp \$

DebugTransformer

author Ben Walding
version \$Id: DebugTransformer.java,v 1.4 2004/03/03 16:19:58 bwalding Exp \$

DeltaTransformer

Compares two sorted streams of data, generating an output stream of INSERTs, UPDATEs and DELETEs.

The DeltaTransformer works by taking two sorted streams of data (using the SortTransformer where required). It then uses an $O(n)$ algorithm to generate a list of deltas.

The input streams must be sorted by the primary key for this transformer to work.

The output of this transformer is as follows

PositionName Description Column 1TYPE The operation (I - Insert, D - Delete, U - Update) Column n * 20[Column Name from first source] The data from column n of the first source Column n * 2 + 11[Column Name from second source] The data from column n of the second source

author Ben Walding
version \$Id: DeltaTransformer.java,v 1.8 2004/03/03 16:19:58 bwalding Exp \$

DifferenceTransformer

FIXME Describe this transformer

property - firstKey = comma delimited list of keys (for first link) that will be used to determine differences
property - firstKey = comma delimited list of keys (for second link) that will be used to determine differences

author Ben Walding
version \$Id: DifferenceTransformer.java,v 1.8 2004/03/03 16:19:58 bwalding Exp \$

DiskCacheTransformer

The CacheTransformer allows complex queries to be stored locally. The CacheTransformer first tries to find a cache file, if it cannot, it opens it's source transformer and passes the data through while writing down to an XML data file.

forge.transformer.property
author Ben Walding
version \$Id: DiskCacheTransformer.java,v 1.14 2004/03/17 06:08:24 bwalding Exp \$

EmptyTransformer

This transformer is an empty transformer. It has no columns and never returns a row.

author Ben Walding
version \$Id: EmptyTransformer.java,v 1.7 2004/03/03 16:19:58 bwalding Exp \$

FilterTransformer

Filters the output of a single source transformer using BeanShell scripts. boolean allow(rs, properties) { filter.expr code here };

author Ben Walding
version \$Id: FilterTransformer.java,v 1.9 2004/03/03 16:19:58 bwalding Exp \$

ForEachTransformer

Joins two transformers together, opening T2 for every record of T1.

The for-each transformer works along the lines of most for-each systems.

You have two transformers T1 / T2.

T1 is the driver query. It is opened once at the beginning of the session and then iterated through. For each record in T1, T2 is closed and then opened using parameters from T1.

This is ideal for situations where you need to join T2 to T1, but cannot afford to open the massive resultset that T2 might represent.

parallelism = n (default 0 = off), parallelism refers to the number of simultaneous child transformers that can be open simultaneously. This can be used in the case where the child is quite slow to open. By setting a high parallelism, they will be opened ahead of time and then used as required. There is an overhead to using parallelism, and if child transformers have side effects between calls, then this could pose problems.

parallelism.sleep=n (default 250), ms to sleep between possibly opening new children. If the queue is exhausted, new ones will be opened immediately.

author Ben Walding
version \$Id: ForEachTransformer.java,v 1.12 2004/05/31 11:50:50 dataforge Exp \$

FormatTransformer

author Ben Walding
version \$Id: FormatTransformer.java,v 1.9 2004/03/11 00:20:31 bwalding Exp \$

GroupByTransformer

author Ben Walding
version \$Id: GroupByTransformer.java,v 1.11 2004/03/03 16:19:58 bwalding Exp \$

HTMLPresentation

author Ben Walding
version \$Id: HTMLPresentation.java,v 1.6 2004/03/03 16:19:59 bwalding Exp \$

JoinTransformer

Joins together N transformers in a star formation.

The JoinTransformer is a transformer for joining transformers in a star formation. Given N tables, T1 .. TN, T2..TN will be cached and joined to T1 on demand

eg. `SELECT T1.*, T2.* FROM T1, T2 WHERE T1.LHS = T2.RHS` LHS / RHS are properties on the transformer

For now, it is assumed that the LHS is the first bind, and it is also the data table. RHS is the PKd (eg. T2.RHS is PK of T2). Data is read from T1 and looked up in T2. All data is retrieved from T2..Tn as the JoinTransformer is opened.

author Ben Walding
version \$Id: JoinTransformer.java,v 1.9 2004/03/03 16:19:58 bwalding Exp \$

MemoryCacheTransformer

Stores the results of the transform in memory to be reused on subsequent openings.

The MemoryCacheTransformer allows complex queries to be stored locally in memory.

author Ben Walding

version \$Id: MemoryCacheTransformer.java,v 1.6 2004/03/03 16:19:58 bwalding Exp \$

MultiplexTransformer

Multiplexes between a group of different transformers.

data-
forge.transformer.propert
multiplex.index Make the multiplexer use the source transformer at the index given
data-
forge.transformer.propert
multiplex.name Make the multiplexer use the source transformer with a link of the
y
given name
author Ben Walding

version \$Id: MultiplexTransformer.java,v 1.8 2004/03/11 01:21:10 bwalding Exp \$

NullTransformer

This transformer does nothing to the data stream.

This is a "do-nothing" pass through transformer. Its primary purpose in life is to provide a convenient default transformer for people to experiment with / use as a default when creating new transformers.

author Ben Walding

version \$Id: NullTransformer.java,v 1.5 2004/03/03 16:19:58 bwalding Exp \$

ObjectPresentation

Ultimately presents the data as a List of Lists

author Ben Walding

version \$Id: ObjectPresentation.java,v 1.6 2004/03/11 00:20:31 bwalding Exp \$

ResultSetJDBCTransformer

Allows a user to query a standard JDBC data source.

data-
forge.transformer.propert
datasource Which datasource should the query be run against
data-
forge.transformer.propert
sql The query to run against the database
author Ben Walding

version \$Id: ResultSetJDBCTransformer.java,v 1.12 2004/03/17 06:08:24 bwalding Exp \$

Retrieves the first N records from its source transformer

data-
forge.transformer.propert
y author max The maximum number of rows to return
Ben Walding
version \$Id: TopNTransformer.java,v 1.7 2004/03/03 16:19:58 bwalding Exp \$

Transformer

Building block of the DataForge, can be opened to get access to JDBC data.

The transformer lifecycle is as follows

constructor

set properties

open() - Returns a worker TransformerResultSet

ready for re-opening as required.

Transformers should not store any state about the resultsets they have provided. There are exceptions such as some caching transformers (eg. MemoryCacheTransformer).

author Ben Walding
version \$Id: Transformer.java,v 1.8 2004/03/11 00:20:31 bwalding Exp \$

XMLPresentation

author Ben Walding
version \$Id: XMLPresentation.java,v 1.8 2004/03/03 16:19:59 bwalding Exp \$

XMLTransformer

For the most part, this transformer is configured from code (and not usually used in the configuration XML files)
XPP - Uses an XML Pull Parser for improved performance and simplicity. The rigorous testing regime should ensure that any changes to optimise the code will not break the rules that have been in place.

data-
forge.transformer.propert
y author source The location of the XML source
Ben Walding
version \$Id: XMLTransformer.java,v 1.11 2004/03/03 16:19:58 bwalding Exp \$

Index

D

DCE, 8

E

expressions
OGNL, 7

O

OGNL expression, 7

T

transformers, 2, 6