GNUe Reporting Concepts

Understanding the GNUe Reports reporting engine.

1. Introduction

GNUe Reports is the primary reporting agent of the GNU Enterprise project. GNUe Reports is designed to extract data from diversified data sources (including all major database vendors) and generate platform- and output-independent reports.

2. Reporting Overview

2.1. XML, XSL, and Output Formats

GNUe Reports uses XML to define reports and as an intermediary format prior to the final output. In other words, GNUe Reports reads in an XML report definition, uses this definition to extract data from a database backend, and generates an XML-based report output. This output can then be transformed into text, HTML, Postscript, PDF, TeX, word processor, Spreadsheet, or a wide variety of other formats. (Currently, HTML and text outputs are directly supported without the use of an intermediary converter.) For other formats, one of the natively supported formats can be piped through another utility (such as ghostscript) to generate a multitude of other formats.

GNUe Reports currently uses XSL Transforms to go from the “report” xml output to the final formats, “text” and “html”. It is not difficult to create new formats by creating a new XSL script. It is also easy to create company-customized reports by modifying a few lines of XSL. In the future, we plan to support “templated” reports so site customizations can be more easily accomplished without the need to know XSL.

2.2. Reporting Model

The two primary components of a report definition are the datasources and the layout directives. The datasources tie the report to a database backend. In a relational database environment, a datasource typically corresponds to a table or a view.

3. Datasources

A datasource contains records and fields of information. Typically it is linked to a defined database and associates itself with a specific source, like a table, in that database. However it is possible to have datasources that have no database link defined. These types of datasources cannot provide persistance to the information stored within themself. Typically, only data-bound datasources will be used in reports.

3.1. Master/Detail Relationships

Datasources can also be associated with other datasoure in what we call a Master/Detail relationship. A master datasource has one or more detail datasources associated with itself. These detail datasources contain a foreign key which points back to a specific master datasource/field key. It's worth noting that a datasource can be both a detail and master datasource simultaneously.

3.2. Record Caching/Prefetch

In an effort to improve system performance datasources will eventually be able to be configured to cache records in memory. The application developer will have control over the number of records cached to allow them to balance memory usage vs system responsiveness. Currently the cache system in GNUe is, in reality, a pre-fetch system. When large numbers of records are requested from the database the system will fetch only the user specified cache amount at a time. When the report moves beyond the number of records in memory, the system will automatically fetch another group of records. This allows the system to respond rapidly to requests involving large number of records.

4. Parameters

Parameters are user-supplied values that can be put in the output stream or used in datasource conditionals.

5. Sorting Options

* *Sorting Options are not fully supported in GNUe Reports. This will be added in an upcoming release.*

Sorting Options are predefined ways to sort a report. These are defined in the Report Definition and

6. Triggers

* *Triggers are not currently supported in GNUe Reports. This will be added in an upcoming release.*

Triggers contain scripts of code that execute during specific events which "fire" the triggers. They can be attached to any of the components of the virtual form though some trigger events may not pertain to the component and, as such, will never fire.

Not implemented yet: Triggers normally return a True value upon sucessful completion. It is possible for the trigger script to return True or False values to influence the applications behaviour.

6.1. Trigger Languages

Currently, trigger scripts must be written in python. We would like to increase the number of supported languages as time permits.

6.2. Trigger Events

The following triggers are either implemented or planned for implementation in the GNUe Forms system. Unless specifically mentioned the return value of the script will not effect the application in any way.

6.2.1. Implemented

No trigger events have been implemented.

6.2.2. Not Implemented

These triggers are not yet implemented. Plans for these triggers may change prior to implementation. Use this section only as a tool to understand our conceptual direction.

Entry

- On-New-Value - Fires when a new value is queried. ?????

Summary

- Pre-Calculate - Fires prior to the summary being calcuated. ?????
- Post-Calculate - Fires following the summary being calcuated. ?????

Section/Layout

- Pre-Process - Fires prior to selecting any records ?????
- On-Select - Fires when a new record is queried. ?????
- Post-Process - Fires after selecting and processing all records ?????

7. Standard Layout Elements

7.1. Overview

7.2. Sections

Sections serve a dual purpose -- to handle logical grouping of fields and ...

7.3. Fields

To reference a field in the current section,

```
<field name="blah" />
```

To reference a field in a higher-level section,

```
<field name="blah" section="blah"/>
```

7.4. Summaries

Summaries provide accumulated statistics for a specific field. This can range from a simple count of the number of occurances of a field, to the total sum or average of this field.

```
<summ field="fieldname" function="sum" section="uppersect"/>
```

Functions:

count	Computes a "count" of the total number of non-empty instances of a field.
sum	Computes a running total of the specified numeric field.
avg	Computes the average of the specified numeric field.
min	Remembers the "smallest" value of a field instance.
max	Remembers the "largest" value in a field instance.

7.5. Parameters

Parameters are pass-through values supplied by the end user. If you create a parameter called “subtitle” and the user specifies “My Tuesday Report” as its value, then any <param name=”subtitle”> will be replaced with the string “My Tuesday Report”. See the chapter on Parameters for more information on defining parameters.

7.6. Formulas

Formulas are currently not supported in GNUe Reports. This will be added in an upcoming release.

Formulas are a simplified version of triggers (or one-shot triggers, so to speak.) Formulas are used to perform a simple calculation with a few fields (e.q., multiply a “quantity” field by the “retail” field to generate an “extended price” value on the fly.)

If you require any added functionality, you more than likely should be using a trigger.

7.7. Layout-level Triggers

Triggers are currently not supported in GNUe Reports. Once they are, this section will describe layout-specific triggers.

8. Layout Templating

8.1. Overview

GNUe Reports, with its versative reporting engine, is not tied to any particular output XML format. In essence, the reporting engine performs a Database-To-XML transformation, with an arbitrary output XML format.

8.2. GNUe Namespaces

For example, assume you have the following layout section:

```
<layout  out:xmlns="GNUe:Reports:SimpleTabulation">
  <out:mygroup>
    <section source="dts1">
      <out:mydata myattr="attrval"><field name="foo"/></out:mydata>
    </section>
  </out:mygroup>
</layout>
```

Furthermore, assume the datasource *dts1* has the following values for field “foo”: *Bob*, *Sam*, and *Jane*. Then the resulting xml will be:

```
<mygroup xmlns="GNUe:Reports:SimpleTabulation">
  <mydata myattr="attrval">Bob</mydata>
  <mydata myattr="attrval">Sam</mydata>
  <mydata myattr="attrval">Jane</mydata>
</mygroup>
```

9. Advanced Topics

9.1. Computing Weighted Averages

This will depend on either formulas or triggers to be supported. When they are supported, you can compute a weighted average by creating a record-level formula that multiplies the two fields containing the weight and the value. Then, you will create a section-level summary that averages (<summ function=”avg”>) this formula-field.

Possible Example:

```
...
<section source="mydts">
   <field name="weight"/>
   <field name="val"/>
   <formula name="avgseed">weight*val</formula>
</section>
<summ field="avgseed" function="avg"/>
...
```

10. GNUe Reports Definition (GRD) File Format

10.1. General Markup

10.1.1. <report>

The <report> tag if the root tag. It’s only purpose is to enclose the report definition. All other tags fall somewhere inside this tag.

Attributes

Attribute	Datatype	Default	Description
title	string	none	The title of the report. Only used when user requests information about a report.
author	string	none	The name of the author of the report.Only used when user requests information about a report.

version	string	none	The specific version number of the report. Only used when user requests information about a report.
description	string	none	A brief description of the report. Only used when user requests information about a report.

Example

```
<report title="Accounting Month End Report"
        author="GNU Enterprise" version="1.2.15">
    <!-- Rest of report logic goes here -->
</report>
```

10.2. User-Supplied Parameters

10.2.1. <parameters>

The parameters tag encloses the parameter definition section. It is a container for <parameter> tags.

Attributes

This tag simply serves as a container. It has no attributes.

Example

```
<parameters>
      <!-- parameter definitions go here -->
</parameters>
```

10.2.2. <parameter>

A parameter tag defines a single user-settable parameter.

Attributes

Attribute	Datatype	Default	Description
name	string	none	A unique identifier for this parameter.

required	boolean	FALSE	Is this parameter required in order for the report to run properly. If “default“ is provided, then this attribute serves no meaning.
limited	boolean	FALSE	This parameter is limited to the results of the specified “source“. The source attribute must be provided in order for this tag to be processed.
default	string	none	The default value of this parameter. This value will be used if the user does not provide a value for this parameter.
description	string	none	A description of this parameter. This should be meaningful to the end-user as this will be used in the prompt for the parameter value when displayeed to the user. If this attribute is not provided, then the name will serve as the prompt.““
source	string	none	Provides a “lookup“ mechanism for this parameter. When “limited“ is set, the value the user supplies must be present in sources. If limited is not set, then the results of sources will be a list of suggested/possible values for the user to select.
type	string	char	Specifies the typecast/data type for this parameter. Should be either char, number, or date.

Note: Currently, the following attributes are not implemented: “required”, “limited”, and “source”. The engine will accept values for these attributes, but does not do any processing of them.

Example

This example defines two parameters, beginDate and endDate, which are both “date” types.

```
<parameters>
   <parameter name="beginDate"
              description="First day of the period"
              type="date"/>
   <parameter name="endDate"
              description="Last day of the period"
              type="date"/>
</parameters>
```

10.3. Sorting Options

NOTE: sortoptions are currently not implemented.

10.3.1. <sortoptions>

The parameters tag encloses the parameter definition section. It is a container for <parameter> tags.

Attributes

This tag simply serves as a container. It has no attributes.

Example

```
<sortoptions>
      <!-- sortoption definitions go here -->
</sortoptions>
```

10.3.2. <sortoption>

A sortoption tag defines a single user-selectable sortoption. See the chapter on Sorting Options for more information on these.

Attributes

TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Attribute	Datatype	Default	Description
title	string	none	The title of the form. Will be displayed on About Screen.
width	int	10	The width of the object in text columns.

height	int	1	The height of the object in text rows.
author	string	none	The name of the author of the form. Will be displayed on About Screen
version	string	none	The specific version number of the form. Will be displayed on About Screen
description	string	none	A brief description of the form to be displayed on About screen.
tabbed	string	none	Allows a form to convert it's pages as notebook tabs. Allowed values are left, right, bottom, top.
requireGUI	boolean	FALSE	NOT IMPLEMENTED YET: If defined the client will abort the form if it is unable to provide graphical display
noTriggerDownload	boolean	FALSE	NOT IMPLEMENTED YET: If defined the client will not attempt to ask the server for triggers to download

Example

```
<report>
  </emphasis>    <!-- Rest of report logic goes here -->
</report>
```

10.3.3. <sortcolumn>

TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Attributes

Attribute	Datatype	Default	Description

title	string	none	The title of the form. Will be displayed on About Screen.
width	int	10	The width of the object in text columns.
height	int	1	The height of the object in text rows.
author	string	none	The name of the author of the form. Will be displayed on About Screen
version	string	none	The specific version number of the form. Will be displayed on About Screen
description	string	none	A brief description of the form to be displayed on About screen.
tabbed	string	none	Allows a form to convert it’s pages as notebook tabs. Allowed values are left, right, bottom, top.
requireGUI	boolean	FALSE	NOT IMPLEMENTED YET: If defined the client will abort the form if it is unable to provide graphical display
noTriggerDownload	boolean	FALSE	NOT IMPLEMENTED YET: If defined the client will not attempt to ask the server for triggers to download

Example

```
<report>
   <!-- Rest of report logic goes here -->
</report>
```

10.4. Layout Elements

10.4.1. <layout>

TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Attributes

Attribute	Datatype	Default	Description
name	string	none	A unique ID for the widget. This is only useful when importing from a library.

Example

<page name=”page1”>

Objects that should be on this page go in here

</page>

<page>

Objects that should be on this page go in here

</page>

10.4.2. <section>

TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Attributes

Attribute	Datatype	Default	Description
name	string	none	A unique ID for the widget.The name of the widget. No blocks or datasources can share the same name without causing namespace collisions in user triggers.
datasource	string	none	The name of a datasource (defined in by a <datasource> tag.) that provides this block with it’s data.

restrictDelete	boolean	none	If set then the user will be unable to request that a record be deleted via the user interface.
restrictInsert	boolean	none	If set then the user will be unable to request that new records be inserted into the block
transparentBlock	boolean	none	If set then the you can tab out of the block via next or previous field events. Makes navigation in mutliblock forms easier.
rows	int	1	Any widgets inside the block will display this number of copies in a verticle column. Simulates a very crude grid entry system. Serves the same purpose as the visibleCount attribute on some widgets.
rowSpacer	int	1	Adjusts the verticle gap of this number of rows between duplicated widgets. Serves the same purpose as some of the gap attributes on individual widgets.

Example

<block name="cities" datasource="city" master="state.st_code" detail="state">

label and entry objects that are part of this block go here

</block>

The cities block defined in the above example is a detail block. It is linked to a datasource named city and it keeps the data displayed within itself in sync with the block named state. It does this by monitoring the entry named st_code in the state block, when that data changes it sets the entry within itself named state to match the value stored in st_code as querying data from the datasource.

<layout>

Attributes

Attribute	Datatype	Default	Description

name	string	none	A unique ID for the widget.The name of the widget. No blocks or datasources can share the same name without causing namespace collisions in user triggers.
datasource	string	none	The name of a datasource (defined in by a <datasource> tag.) that provides this block with it’s data.
restrictDelete	boolean	none	If set then the user will be unable to request that a record be deleted via the user interface.
restrictInsert	boolean	none	If set then the user will be unable to request that new records be inserted into the block
transparentBlock	boolean	none	If set then the you can tab out of the block via next or previous field events. Makes navigation in mutliblock forms easier.
rows	int	1	Any widgets inside the block will display this number of copies in a verticle column. Simulates a very crude grid entry system. Serves the same purpose as the visibleCount attribute on some widgets.
rowSpacer	int	1	Adjusts the verticle gap of this number of rows between duplicated widgets. Serves the same purpose as some of the gap attributes on individual widgets.

Example

<block name="cities" datasource="city" master="state.st_code" detail="state">

label and entry objects that are part of this block go here

</block>

The cities block defined in the above example is a detail block. It is linked to a datasource named city and it keeps the data displayed within itself in sync with the block named state. It does this by monitoring the entry named st_code in the state block, when that data changes it sets the entry within itself named state to match the value stored in st_code as querying data from the datasource.

10.4.3. <field>

TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Attributes

Attribute	Datatype	Default	Description
x	int	none	The text column starting position of the widget . Based upon leftmost column of screen being 0.
y	int	none	The text row starting position of the widget. Based upon the top row of the screen being 0.
text	string	none	The text to be displayed.
width	int	10	The width of the label in text columns. Defaults to the width of the text. Only really usefull when used with the justification attribute.
alignment	string	left	The justification of the label. Can be one of the following left, right, or center. Requires that the width attribute be set.
rows	int	1	Overrides the rows setting defined at the block level.
rowSpacer	int	1	Overriders the rowSpace setting defined at the block level.
name	string	none	The unique ID of the label.

Example

10.4.4. <summ>

TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Attributes

Attribute	Datatype	Default	Description
x	int	none	The text column starting position of the widget . Based upon leftmost column of screen being 0.
y	int	none	The text row starting position of the widget. Based upon the top row of the screen being 0.
text	string	none	The text to be displayed.
width	int	10	The width of the label in text columns. Defaults to the width of the text. Only really usefull when used with the justification attribute.
alignment	string	left	The justification of the label. Can be one of the following left, right, or center. Requires that the width attribute be set.
rows	int	1	Overrides the rows setting defined at the block level.
rowSpacer	int	1	Overriders the rowSpace setting defined at the block level.
name	string	none	The unique ID of the label.

Example

10.4.5. <param>

TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Attributes

Attribute	Datatype	Default	Description
x	int	none	The text column starting position of the widget . Based upon leftmost column of screen being 0.
y	int	none	The text row starting position of the widget. Based upon the top row of the screen being 0.
text	string	none	The text to be displayed.
width	int	10	The width of the label in text columns. Defaults to the width of the text. Only really usefull when used with the justification attribute.
alignment	string	left	The justification of the label. Can be one of the following left, right, or center. Requires that the width attribute be set.
rows	int	1	Overrides the rows setting defined at the block level.
rowSpacer	int	1	Overriders the rowSpace setting defined at the block level.
name	string	none	The unique ID of the label.

Example

10.4.6. <default>

TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Attributes

This tag has no attributes.

Example

10.4.7. <firstRow>

TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Attributes

This tag has no attributes.

Example

10.4.8. <notFirstRow>

TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Attributes

This tag has no attributes.

Example

10.4.9. <lastRow>

TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Attributes

This tag has no attributes.

Example

10.4.10. <notLastRow>

TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Attributes

This tag has no attributes.

Example

11. The *grrun* Command-line Client

The grrun utility is currently the only interface to the GNUe Reports engine.

11.1. Command line options (switches)

The following options are available. (This list is also available by running “grrun --help”

--comment, -C Include structural comments in the XML output stream. Useful for debugging .grd files.

--connections <loc> Specifies the location of the connection definition file. <loc> may specify a file name (/usr/local/gnue/etc/connections.conf), or a URL location (http://localhost/connections.conf). If this option is not specified, the environent variable GNUE_CONNECTIONS is checked.

--debug-file <file> Sends all debugging messages to a specified file (e.g., "--debug-file trace.log" sends all output to "trace.log") *[for developer's use]*

--debug-level <level> Enables debugging messages. Argument specifies the level of messages to display (e.g., "--debug-level 5" displays all debugging messages at level 5 or below.) *[for developer's use]*

--destination <dest>, -d Where should the report be output to? The value of this depends on the destination type (e.g., if sending to printer, then -d specifies the printer name; if sending via email, then -d specifies the email address.) If <dest> is "-", then output is sent to stdout -- NOTE: when sending to stdout, also use the -q [--quiet] option or you may get junk in your output stream. NOTE: Currently the default value is "-" -- this may change once GNUe Reports is formally released!

--destination-options <opts> Options to pass to the destination process. Available options are specific to the type of destination. Example: '--destination-options "-o nobanner" '

--destination-type <type>, -D This specifies how the report should be output. The currently supported values for <type> are file [default], printer, email, and fax. Note that printer, email, and fax are sent via the server's machine, not the client's machine. To NOTE: Only file, printer, and email are currently implemented!

--exclude-xml, -X Do not output GNUe Reports runtime XML markup information. If specified, then the GRDs layout section will be processed and output as-is; i.e., without any additional information added by GNUe Reports.

--filter <filt>, -f Select the filter to be used to process report output. <filt> is the name of the filtering process as defined on the Report Server machine. If not specified, the "raw" filter is used (i.e., no filtering takes place.)

--filter-list List the available [predefined] filters available to GNUe Reports

--filter-options <opts>, -F Options to pass to the filter process. Available options are specific to the filter. --filter-list will list available filters and their options. Example: '--filter-options "paper=letter margin=1,1,1,1" '

--help Displays this help screen.

--*interactive-debugger* Run the app inside Python's built-in debugger *[for developer's use]*

--pass <passwd> Password used to log into the database. Note that if specified, this will be used for all databases. If not supplied, the program will prompt for password. NOTE: SUPPLYING A PASSWORD VIA THE COMMAND LINE MAY BE CONSIDERED A SECURITY RISK.

--profile Run Python's built-in profiler and display the resulting run statistics. *[for developer's use]*

--quiet, -q Run GNUe Reports in quiet mode -- i.e., display no output. NOTE: if --debug-level is specified, then suppressed text will be output as debugging information (at debug level 1)

--sort <sort>, -s Select the "sort-option" used to sort the report.

--standalone, -S Create a standalone, single-use server instance. Use this option in a non-client/server environment or in a debugging/development environment. NOTE: Until the Reports Server is operational, this mode is implied

--user <name>, -u Username used to log into the database. Note that if specified, this will be used for all databases. If not supplied, the program will prompt for username.

--version Displays the version information for this program.