ReportEngine developer’s guide

Version

Contents

[Writing a custom data column 3](#_Toc426801620)

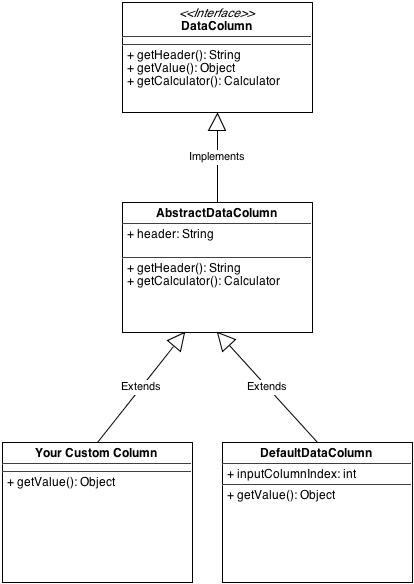
[Writing a custom input 5](#_Toc426801621)

[Writing a custom output 5](#_Toc426801622)

[Writing a custom calculator 5](#_Toc426801623)

# Writing a custom data column

As you probably remember, report-engine doesn’t support only DefaultDataColumns. There’s a class hierarchy supporting the columns feature.



If you don’t want a special column you can always implement the [DataColumn](http://reportengine.sourceforge.net/html/apidocs/net/sf/reportengine/config/DataColumn.html) interface or, even better, extend the [AbstractDataColumn](http://reportengine.sourceforge.net/html/apidocs/net/sf/reportengine/config/AbstractDataColumn.html) abstract class. Let’s assume you want to append the values of two other columns into one. There no default implementation for this behavior but this is where you can extend the framework and add your custom implementation. Let’s assume you have the following 4 columns input:

|  |  |  |  |
| --- | --- | --- | --- |
| Bonus | FirstName | Salary | LastName |
| 100 | john | 2000 | doe |
| 170 | jack | 1250 | ripper |
| 220 | tom | 1340 | jones |
| 34 | bill | 254 | clinton |

… but you need a list of only two columns where the first contains the full name (i.e. 2nd + 4th column) and the second column contains the sum of the first and the third

First let’s see the code for the full name (2nd + 4th column values)

**import** net.sf.reportengine.config.AbstractDataColumn;

**import** net.sf.reportengine.core.algorithm.NewRowEvent;

/\*\*

\* This custom column contains the appended values

\* of the second and the fourth columns

\*/

**public** **class** FullNameCustomDataColumn **extends** AbstractDataColumn {

/\*\*

\* Constructor for full name data column

\* **@param** header

\*/

**public** FullNameCustomDataColumn(){

**super**("Full Name");

}

@Override

**public** Object getValue(NewRowEvent newRowEvent) {

Object[] inputRow = newRowEvent.getInputDataRow();

String secondColumnValue = (String)inputRow[1];

String fourthColumnValue = (String)inputRow[3];

//append the second and fourth column values

**return** secondColumnValue + " " + fourthColumnValue;

//this is just for instructional purposes.

//One should use a String Builder/Buffer for such operations

}

}

You can find the code [here](http://svn.code.sf.net/p/reportengine/code/trunk/reportengine-samples/src/main/java/net/sf/reportengine/samples/flat/customColumns/FullNameCustomDataColumn.java).

The [SumCustomColumn](http://svn.code.sf.net/p/reportengine/code/trunk/reportengine-samples/src/main/java/net/sf/reportengine/samples/flat/customColumns/SumCustomColumn.java) is almost the same only that returns an integer. More important is the code for of the report itself:

/\*\*

\* this report uses a custom column (full name column)

\*/

**public** **class** CustomColumnFlatReport {

**public** **static** **void** main(String[] args) {

FlatTable flatTable = **new** FlatTable.Builder()

.input(**new** TextTableInput("names.txt","\t"))

.addDataColumn(**new** FullNameCustomDataColumn())

.addDataColumn(**new** SumCustomColumn())

.build();

…

}

}

The code above can be found [here](http://svn.code.sf.net/p/reportengine/code/trunk/reportengine-samples/src/main/java/net/sf/reportengine/samples/flat/customColumns/CustomColumnFlatReport.java)

The final output will look like:

|  |  |
| --- | --- |
| My custom columns report | |
| **Full-name** | **Full Salary** |
| john doe | 2100 |
| jack the ripper | 1420 |
| tom jones | 1560 |
| bill clinton | 2880 |

# Writing a custom input

Section under construction

# Writing a custom output

Section under construction

# Writing a custom calculator

Section under construction