A ledger-sheet maker for Java

Reports.jar

Programmer’s Manual

3th edition

July 27. 2011

Pao@Office

Pao@office Ltd.
3-29-2-401　Yatsu Narashino, Chiba
Zip Code 275-0026

http://www.pao.ac/en/

Table of Contents

Introduction

Hello all programmers who create program under the .NET developing environment.
The interface of Reports.NET as a class is so simple and easy that it requires little effort.
*It means that a report definition XML file as a design part plays a major role.
There are a few classes and methods such as:

```
IReports Interface---Common interface for print or preview
 │
 ├── loadDefFile Method    ---Read a report definition file
 │
 ├── pageStart Method      ---Declare a start of a page
 │
 ├── write Method          ---Write print data
 │
 ├── pageEnd Method        ---Declare an end of a page
 │
 ├── output Method         ---Order print / preview
 │
 ├── loadXMLFile Method --- Read print data file
 │
 ├── saveXMLFile Method --- Write print data file
 │
 ├── saveData Method        ---Return compressed print binary data
 │                                  (Transfer format with Web service)
 ├── loadData Method        ---Read compressed print binary data
 │                                  (Transfer format with Web service)
 │
 ├── savePdf Method         ---Save PDF format in directed format
 │
 └── savePdfData Method    ---Return PDF Data to binary data

ReportCreator Class     ---Return preview instance(object)
 │          (IReports type of the above)
 └── getReport Method        ---Return print object
```

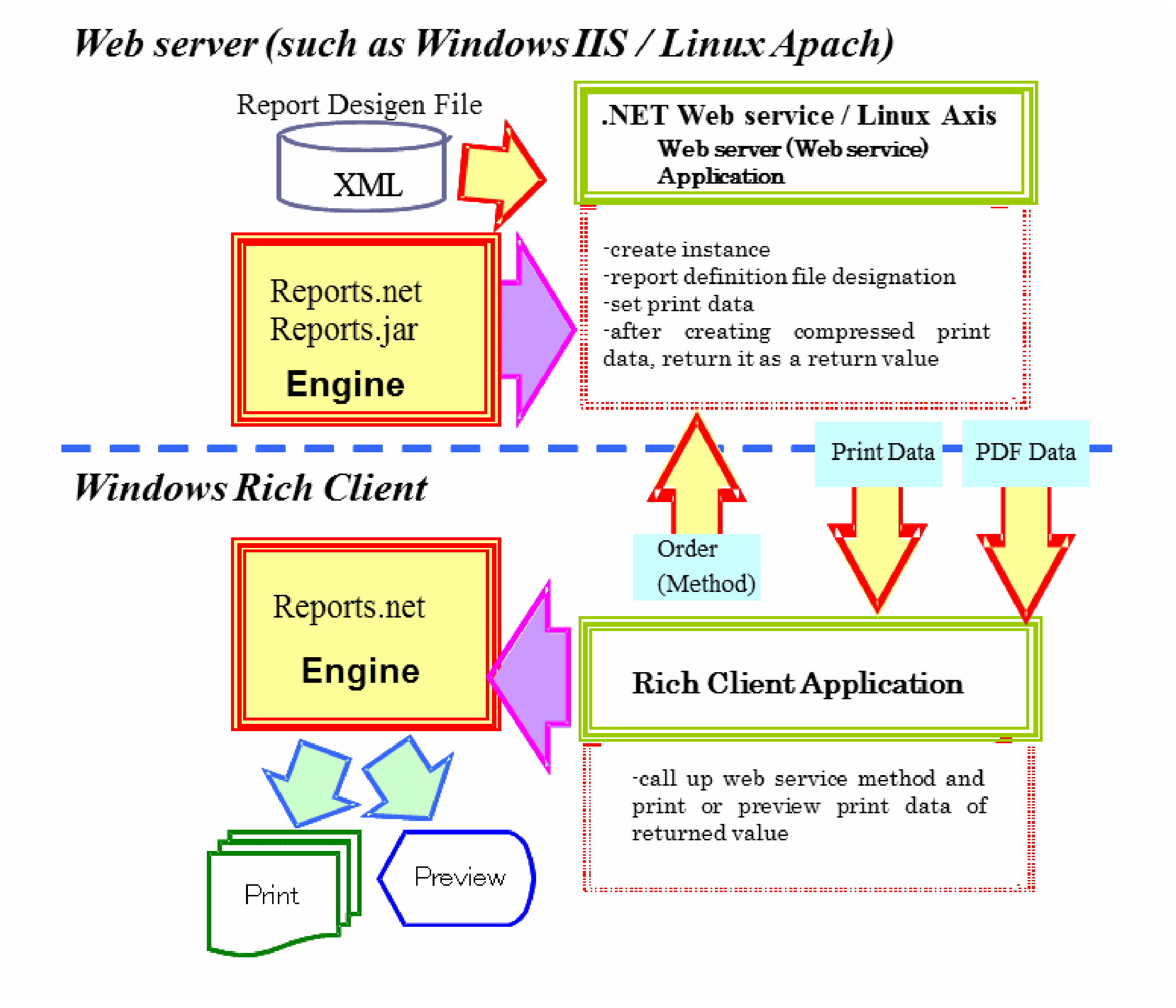
That's it.　That's all you need.
*You will also see ReportStartImpl class but please don't worry about it.　It is only for preview reboot.
What do you think?　It looks like nothing to worry about.
Now let's move on to the details of each class and method with some examples such as coding.
I sincerely hope that all programmers will enjoy programming easily with this software.

Creator

Functions

As soon as you order one to (call method for) WEB service (kinds of axis) on UNIX (Linux) sever from Rich Client Windows as Platform, you will obtain binary data which is compressed via WEB server and you can print.

When you are ordered from client (is called method), Server access databases and crate print data, after Return the data to Client as a binary data (a variable of type byte) ,Client prints it. Furthermore, if you have not Reports.NET, you can output ledger sheet as PDF due to being able to output PDF Data by Reports.jar.


Web server (such as Windows IIS / Linux Apach)
Report Desigen File
XML
.NET Web service / Linux Axis
Web server (Web service) Application
Reports.net
Reports.jar
Engine
-create instance
-report definition file designation
-set print data
-after creating compressed print data, return it as a return value
Print Data
PDF Data
Windows Rich Client
Order (Method)
Reports.net
Engine
Rich Client Application
-call up web service method and print or preview print data of returned value
Print
Preview

Operating Condition

In order to use this software, a computer which meets the following requirements is needed.

OS	Systems that operate at least JDK1.4 sufficiently
Computer Memory	Equivalent to the memory allocation which at least JDK1.4 to operate sufficiently
Recommended Screen Resolution	No special limit
Developing Environment	We recommend you to install eclipses　(＾_＾;)

How to Use

You have to only copy Pao.Reports.jar from.

http://www.pao.ac/en/reports.net/

Please download and install Reports.net.

The installation folder of Reports.net, there are variety files of Reports.jar.

When you use eclipse, add Pao.Reports.jar with build path archive.

1. How to use Reports.jar from application program

Sample Program as an Example

In all this section, How to Use Reports.jar from Apprication Program, explanations will be made with sample programs.　Please keep the following in mind.

＜Preprocessing in Client program Windows ＞

- This program is asked from Client side .Net program via Web Services (axis).

＜Program Descriptions＞

- Write the date and page numbers to each ledger sheet.
- Descriptions is written the numbers of lines looped 60 times and the decuple values of the each number in a chart.
- Each line of the detail part will be separated by horizontal ruled lines.
- A page will be broken with 15 lines so there will be 4 pages all together

Finally, save the print data which was previously printed or previewed to a print data file, reread the file again and preview the print data.

＜Post-processing in Client Program on Windows＞

- After Windows client received the binary print data, you will print or preview the data

The sample program which performs the above process is made and offered for reference. The sample also has some comment and they would be helpful.

Please keep the processing flow of the sample program in mind.

This sample program is found in the folder, sample¥programers, in a compressed file of the product.

```
public byte[] getBaisuu()
{
	byte[] ret = null;

	//Creating Instance
	IReport paoRep = ReportCreater.getReport();

	try
	{
		//Read definition ledger sheets
		paoRep.loadDefFile
		("/usr/local/tomcat/webapps/axis/WEB-INF/classes/pao/Programers.xml");

		int page = 0; // Define the number of pages
		int line = 0; // Define the number of lines

		for (int i = 0; i < 60; i++)
		{
			if (i % 15 == 0) // Start a page with 15 lines
			{
				// Declare a star of a page
				paoRep.pageStart();
				page++;          // Increment the number of pages
				line = 0;       // Reset the number of lines

				//***Header setting***
				// Set character strings
				GregorianCalendar cal = new GregorianCalendar();
				paoRep.write("DateTime", cal.getTime().toString());
				paoRep.write("Page", "Page - " + Integer.toString(page));

			}
			line++; // Increment the number of lines


			//***Detail setting***
			// Set repeated character strings
			paoRep.write("LineNo", Integer.toString(i+1) , line);
			paoRep.write("10Baisu", Integer.toString((i+1)*10) , line);
			// Set repeated figure (horizontal line)
			paoRep.write("HLine", line);

			if (((i+1) % 15) == 0) paoRep.pageEnd(); // One page, page breaks in the line 15
		}

		ret = paoRep.saveData(); // Save print data
	}
	catch(Exception ex)
	{}

	return ret;

}
```

Read Report Definition File

When you set data to a ledger sheet from a program, the first step is to read a report definition file such as the one created with Designer.
*Definitions for a ledger sheet, including which coordinate has which object, are written in XML file format in report definition file. For details, please refer to the Report Definition XML File Specification Document.

Use the loadDefFile method to read a report definition file from a program implemented in IReport interface. Set the path of the report definition file which is to be read to the first argument of loadDefFile method.

＜Example＞

```
//Read report definition file
paoRep.loadDefFile
("/usr/local/tomcat/webapps/axis/WEB-INF/classes/pao/Programers.xml");
```

Declare Start and End of a Page

When you set date to a ledger sheet from a program, a report definition file should be read and declaration of start and end at each page should be made.
Set the ledger sheet data between the declaration of the start and the end. There is no need for the data setting when outputting the ledger sheet as showed in the report definition file created by Designer.

In other words, the minimum steps of outputting a ledger sheet by reading a report definition from a program are:

1. Create a print or a preview instance.
2. Read a report definition file.
3. Declare a start of a page.
4. Declare an end of the page.
5. Set print data as a variable of type byte, and return to client .

In most cases, the logic is inserted which set a ledger sheet data between "3. Declare a start of a page" and "4. Declare an end of the page".
To declare start and end of a page, use pageStart/pageEnd method implemented in IReport interface.
There is no argument.

＜Example＞

```
// Declare a star of a page
paoRep.pageStart();
        ⋮        Write() ---print data processing
// Declare a end of the page
paoRep.pageEnd();
```

Data Set for an Object

This section explains, how to put a value to each object designated by report definition file and how to draw horizontal ruled lines in a chart repeatedly.　Data set for an object should be created between the start and the end declarations of the page (between pageStart and pageEnd).

When data is set from a program to a ledger sheet, use the write method implemented in IReport interface.　The write method implements three patterns.

(1) void Write(string name, string value)

This sets a character string to an object.

Use this to set a value of unrepeated persistent objects such as header and footer.

string name

This specifies a name of an object in report definition file.

The intentded （intended） object types are Text (character string), ArtText (decorated character string) and Barcode (barcode) because this sets character string.

String value

This specifies a character string to set.

(2) void write(String name, String value, int index)

This specifies a drawing position for an object and set a character string.

Use this for objects to set repeated values such as lines in a chart.

When using the method for this pattern, IntervalX or IntervalY in the report definition file should be entered with a value more than 1.

IntervalX means intervals repeated in a transverse direction by milimeter.

IntervalY means intervals repeated in a longitudinal direction by millimeter. This is mainly used for lines of a chart.

String name

This specifies a name of an object in report definition file.

The intended object types are basically Text (character string), ArtText (decorated character string) and Barcode (barcode) because this sets character string.

String value

This specifies a character string to set.。

int index

This refers to the drawing position in the page set in a transverse and a longitudinal direction intervals by IntervalX／IntervalY. The values get bigger from upper left to lower right.

For example, a chart with a value in IntervalY has a drawing position such as:

(the first position of an object) + InterbalY × (index –1).

As for a chart, 1 is in the first line, 2 in the second and 3 in the third.

void write(String name, int index)

void Write(string name, int index)

This specifies a drawing position for an object. Use this for objects to set repeated values such as horizontal ruled lines in a chart.

When using the method for this pattern, IntervalX or IntervalY in report definition file should be put with a value more than 1.

IntervalX means intervals repeated in a transverse direction by milimeter.

IntervalY means intervals repeated in a longitudinal direction by millimeter. This is mainly used for lines of a chart.

String name

This specifies a name of an object in report definition file.

All objects can be applied because any objects can be drawn repeatedly.

int index

This means printing position in the page set in a transverse and a longitudinal direction intervals by IntervalX／IntervalY. The values get bigger from upper left to lower right.

For example, a chart with a value in IntervalY has a painting position such as: (the first position of an object) + InterbalY × (index –1).

As for a chart, 1 is in the first line, 2 in the second and 3 in the third.

＜Example＞

```
int page = 0; // Define the number of pages
int line = 0; // Define the number of lines
for (int i = 0; i < 60; i++)
{
            if (i % 15 == 0) // Start a page with 15 lines
            {
                        // Declare a star of a page
                        paoRep.PageStart();
                        page++;              // Increment the number of pages
                        line = 0;  // Reset the number of lines

                       //***Header setting***
                       // Header setting
                       GregorianCalendar cal = new GregorianCalendar();
                       paoRep.write("DateTime", cal.getTime().toString());
                       paoRep.write("Page", "Page - " + Integer.toString(page));
            }
            line++; // Increment the number of lines

             //***Detail setting***
             // Set repeated character strings
             paoRep.write("LineNo", Integer.toString(i+1) , line);
             paoRep.write("10Baisu", Integer.toString((i+1)*10) , line);
             // Set repeated figure (horizontal line)
             paoRep.write("HLine", line);

            if (((i+1) % 15) == 0) paoRep.PageEnd(); // Declare the end of the page with 15
}
```

Obtain Compressed Print Binary Data

In order to obtain a compressed print binary data from a program, use SaveData method implemented in IReport interface.　There is no argument.
Byte[] type compressed data is contained in Return Value

Programmer’s Reference

IReport Interface

IReport Interface is the interface which has all methods controlling Reports.NET.
It is possible to create instance using getPreview method or getReport method in reportCreator class. Create instance with getPreview for previewing and with getReport for printing.

Constructor
There is no argument.

Public Method

loadDefFile	Read a report definition file
pageStart	Declare a start of a page
pageEnd	Declare an end of a page
write	Write print data
saveXMLFile	Write a print data file
saveData	Return compressed print binary data
savePdf	Save PDF format data in directed file
savePdfData	Return PDF format data to binary data

ReportCreator Class

ReportCreator class is implemented with a method which returns object to print or preview.

This contains IReport type of getReport method.

Call getPreview method for previewing

Public Method

getReport	Return a print object.

getReport Method

This method is to return object controlling print.
Firstly, perform this method because of creating instance of print object.

＜Example＞
IReport GetReport()

```
//Obtain instance of print object
paoRep = ReportCreator.GetReport();
```

Reference

ReportCreator class

loadDefFile Method

A report definition file is read with the loadDefFile Method.
We recommend the absolute path because it is not sure where a program would run.

＜Example＞

paoRep.loadDefFile(String name)

String name

report definition file

```
//Read report definition file
paoRep.loadDefFile
("/usr/local/tomcat/webapps/axis/WEB-INF/classes/pao/Programers.xml");
```

Reference

IReport interface

pageStart Method

Declare a start of a page with pageStart Method.
Put a code to set print data between start and end (pageEnd) declarations of a page.

<Example>
void pageStart()

```
// Declare a star of a page
paoRep.pageStart();
        ⋮
// Declare the star of the page
paoRep.pageEnd();
```

write() ---print data processing

Reference

IReport interface

pageEnd Method

PageEnd Method

Declare an end of a page with pageEnd Method.
Put a code to set print data between start (pageStart) and end declarations of a page

＜Example＞
void pageEnd()

```
// Declare a star of a page
paoRep.pageStart();
        ⋮          write() ---print data processing
// Declare the end of the page
paoRep.pageEnd();
```

Reference

IReport interface

Write Method

With Write Method, operate object specified by report definition file, such as writing characters or drawing horizontal ruled lines repeatedly to the object specified by report definition file.

List of Overload

void write(String name, String value)

This sets a character string to an object. Use this to set a value of unrepeated persistent objects such as header and footer.

void write(String name, String value, int index)

This specifies a drawing position for an object and set a character string.
Use this for objects to set repeated values such as lines in a chart.

void write(String name, int index)

This specifies a drawing position for an object
Use this for objects to set repeated values such as horizontal ruled lines in a chart.

Reference

IReport interface

void write(String name, String value) Method

This sets a character string to an object.

Use this to set a value of unrepeated persistent objects such as header and footer.

String name

This specifies a name of an object in report definition file.

The intentded (intended) object types are basically Text (character string), ArtText (decorated character string) and Barcode (barcode) because this sets character string.

String value

This specifies a character string to set.

＜Example＞

```
// Set character strings
paoRep.write("DateTime", cal.getTime().toString());
```

Reference

IReport interface

void write(String name, String value, int index) Method

This specifies a drawing position for an object and set a character string.
Use this for objects to set repeated values such as lines in a chart.
When using the method for this pattern, IntervalX or IntervalY in report definition file should be put with a value more than 1.
IntervalX means intervals repeated in a transverse direction by milimeter.
IntervalY means intervals repeated in a longitudinal direction by millimeter. This is mainly used for lines of a chart.

String name
This specifies a name of an object in report definition file.
The intended object types are basically Text (character string), ArtText (decorated character string) and Barcode (barcode) because this sets character string.
String value
This specifies a character string to set.

int index
This means drawing position in the page set in a transverse and a longitudinal direction intervals by IntervalX／IntervalY. The values get bigger from upper left to lower right.
For example, a chart with a value in IntervalY has a drawing position such as:
(the first position of an object) + InterbalY × (index –1).
As for a chart, 1 is in the first line, 2 in the second and 3 in the third.

＜Example＞

```
// Set repeated strings
paoRep.write("Page", "Page - " + Integer.toString(page));
```

Reference

IReport interface

void write(String name, int index) Method

This specifies a drawing position for an object
Use this for objects to set repeated values such as horizontal ruled lines in a chart.
When using the method for this pattern, IntervalX or IntervalY in report definition file should be put with a value more than 1.
IntervalX means intervals repeated in a transverse direction by milimeter.
IntervalY means intervals repeated in a longitudinal direction by millimeter.　This is mainly used for lines of a chart.

String name
　This specifies a name of an object in report definition file.
　All objects can be applied because any objects can be drawn repeatedly.
int index
　This means printing position in the page set in a transverse and a longitudinal direction intervals by IntervalX／IntervalY.　The values get bigger from upper left to lower right.
　For example, a chart with a value in IntervalY has a printing position such as: (the first position of an object) + InterbalY　×　(index –1).
　As for a chart, 1 is in the first line, 2 in the second and 3 in the third.

＜Example＞

```
line++; // Increment the number of lines

//***Detail setting***
// Set repeated strings
paoRep.write("LineNo", Integer.toString(i+1) , line);
paoRep.write("10Baisu", Integer.toString((i+1)*10) , line);
// Set repeated figure (horizontal line)
paoRep.write("HLine", line);
```

Reference
IReport interface

saveData Method

This returns compressed print binary data.
Use this method to create print data to be returned to a rich client at Web service.

＜Example＞
byte[] saveData()

```
byte[] b = paoRep. saveData(); // Return compressed print binary data
```

Reference
IReport interface

saveXMLFile Method

This method saves print data to XML file.

＜Example＞
bool saveXMLFIle (String name)
String name
    Print data XML file path name to be saved

```
paoRep.saveXMLFile("print data.XML"); //Save print data
```

Reference
IReport interface

savePdf Method

Save PDF format data in directed file.
This method is used to output PDF file to browsers

＜Example＞
bool savePdf(String name)
String name
    Save PDF file path name

```
paoRep.savePdf ("print data.pdf"); //Save PDF print data
```

Reference

IReport interface

savePdfData Method

Return PDF format data to binary data.
This method is used to return Rich client print PDF data by WEB service side.リ

＜Example＞
byte[] savePdfData()

```
byte[] b = paoRep.savePdfData(); // Return PDF binary data
```

Reference

IReport interface

Modification History

Version	Release Date	Modification
1	Aug 01 2006	New Release
2	Oct 10 2008	Addition of PDF Output Method savePdf / savePdfData
3	Jul 27 2011	English Version Release