Java 用帳票作成ツール

# Reports.jar

# プログラマーズマニュアル

第 1 版
2006 年 8 月 1 日

Pao@Office

有限会社パオ・アット・オフィス
郵便番号 275-0026
千葉県習志野市谷津 3-29-2-401

http://www.pao.ac/

# 目次

## はじめに

Java 開発環境下で開発を行っているプログラマの皆様、こんにちは、お疲れ様です。
Reports.jar の クラスとしてのインターフェースは、非常に単純で簡単なので本書を読むにあたって心してかからなくて良いです。
※その分、デザイン部であるレポート定義 XML ファイルに頼るところが多いということです。
クラスやメソッドの数も少ししかありません。
もう、ここであげてしまいましょうか。

IReports インタフェース ･･･印刷又はプレビューを行うための共通インターフェース

- loadDefFile メソッド ･･･レポート定義ファイルを読み込む
- pageStart メソッド ･･･ページの開始だよ、と宣言する
- pageEnd メソッド ･･･ページの終了だよ、と宣言する
- write メソッド ･･･印刷データを書き込む
- saveData メソッド ･･･圧縮した印刷バイナリデータを返す
- saveXMLFile メソッド ･･･印刷データファイルを書き出す

ReportCreator クラス ･･･印刷又は、プレビューのインスタンス(オブジェクト)を返す
(上記の IReports 型)

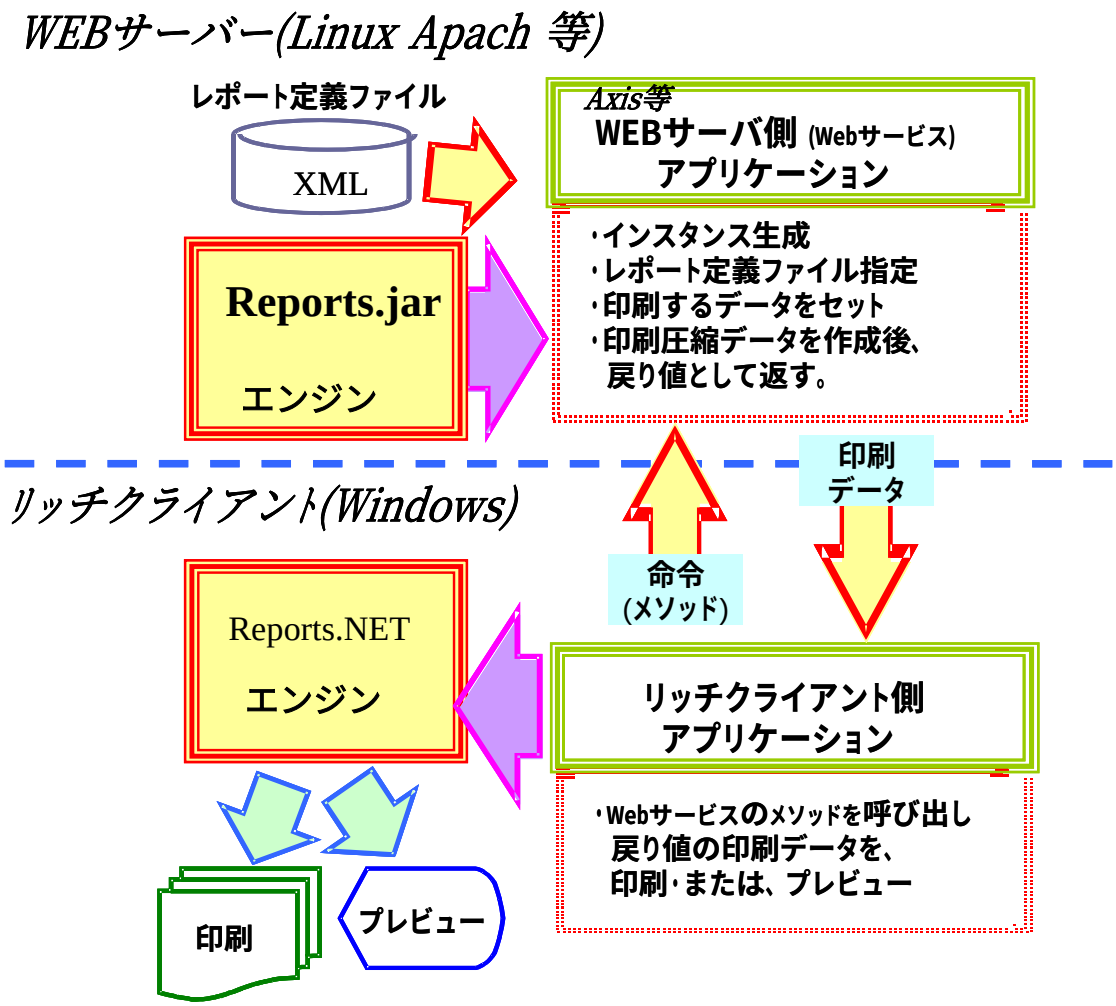
- getReport メソッド ･･･印刷オブジェクトを返す

たったこれだけです。たったこれだけなんです。
どうです？たいしたことないでしょう。
それでは本書内ではコーディング例等を用いながら、各クラスやメソッドについてもう少し細かく書いていくことにします。
皆様が楽しんで楽にプログラミングできることを心から願います。

作者

## 機能概要

Windows をプラットフォームとしたリッチクライアントから、UNIX(Linux 等)サーバ上の WEB サービス(axis 等)に対して 1 つの命令を下す(メソッドを呼び出す)だけで、WEB サーバから印刷データを圧縮したバイナリデータを取得し、印刷を行うことが可能です。クライアントから命令がきたら(メソッドが呼び出されたら)サーバ側のみでデータベース等にアクセスして印刷データを作成し、バイナリデータ(byte[]型変数)として、クライアント側に返し、クライアント側でそれを印刷するという仕組みです。



## 動作条件

本製品を使用するためには、以下の条件を満たす環境のパソコンが必要です。

| | |
|---|---|
| OS | JDK1.4 以上が正常に動作するものである事 |
| 動作に必要なメモリ | JDK1.4 以上が正常に動作するために必要な容量 |
| 画面解像度 | 特に制限なし |
| 開発環境 | eclipse 等 がインストールされていれば、なおよし。(^_^;) |

## 使用方法

Pao.Reports.jar をコピーするだけです。

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

より、Reports.NET(バージョン).zip をダウンロードして解凍後、任意のフォルダにコピーしてください。

eclipse を使用する場合、「ビルドパスー外部アーカイブの追加」で、Pao.Reports.jar を追加してください。

1. アプリケーションプログラムからの Reports.jar 使用方法

**例題サンプルプログラムの紹介**

「アプリケーションプログラムからの Reports.jar 使用方法」全般では、ここに示す例題サンプルプログラムにそって都度説明をしていきます。まず、大体のプログラムの流れを頭に入れておいて下さい。

＜Windows クライアントプログラム前処理＞

- まず、前提として、クライアントの.NET プログラムから、Web サービス(axis 等)経由で、このプログラムが呼び出されます。

**＜本プログラムの説明＞**

- 帳票の各ページのヘッダに日時と頁数を書き込みます。
- 明細部は、６０回ループしてその行番号と、回数を１０倍した値を表に書き込みます。
- 明細部の各行は、横罫線で区切られます。
- 改ページの条件は１５行なので、全部で４ページになります。

以上の描画が済むと、印刷データをバイト方変数に保存し、復帰します。

＜Windows クライアントプログラム後処理＞

- Windows クライアントは、その後、受け取ったバイナリ印刷データを、プレビューまたは、印刷します。

以上の処理を実現しているサンプルプログラムを作成しましたので、参考までに少し追ってみてください。コメントが入っておりますので、そこを読むだけでも構いません。

ここでは、サンプルプログラムの処理の流れを頭に入れておいてください。

なお、このサンプルプログラムは、本製品の圧縮ファイルの中に納められています。

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

    //インスタンスの生成
    IReport paoRep = ReportCreater.getReport();

    try
    {
        //帳票定義体の読み込み
        paoRep.loadDefFile
        ("/usr/local/tomcat/webapps/axis/WEB-INF/classes/pao/Programers.xml");

        int page = 0; //頁数を定義
        int line = 0; //行数を定義

        for (int i = 0; i < 60; i++)
        {
            if (i % 15 == 0) //1頁15行で開始
            {
                //頁開始を宣言
                paoRep.pageStart();
                page++;      //頁数をインクリメント
                line = 0;    //行数を初期化

                //＊＊＊ヘッダのセット＊＊＊
                //文字列のセット
                GregorianCalendar cal = new GregorianCalendar();
                paoRep.write("DateTime", cal.getTime().toString());
                paoRep.write("Page", "Page - " + Integer.toString(page));

            }
            line++; //行数をインクリメント

            //＊＊＊明細のセット＊＊＊
            //繰返し文字列のセット
            paoRep.write("LineNo", Integer.toString(i+1) , line);
            paoRep.write("10Baisu", Integer.toString((i+1)*10) , line);
            //繰返し図形(横線)のセット
            paoRep.write("HLine", line);

            if (((i+1) % 15) == 0) paoRep.pageEnd(); //1頁15行で終了宣言
        }

        ret = paoRep.saveData(); // 印刷データを保存
    }
    catch(Exception ex)
    {}

    return ret;

}
```

## レポート定義ファイル読み込み方法

プログラムから帳票にデータをセットする場合などは、デザイナー等で作成されたレポート定義ファイルをまず読み込みます。

※レポート定義ファイルには、どの座標にどのオブジェクトがある等、帳票の定義が XML ファイル形式で書き込まれております。詳しくは、「レポート定義 XML ファイル仕様書」を参照してください。

プログラムからレポート定義ファイルを読み込むには、IReport インタフェースに実装されている loadDefFile メソッドを使用します。loadDefFile メソッドの第一引数に読み込むレポート定義ファイルのパスを指定してください。

＜例＞

```
//帳票定義体の読み込み
paoRep.loadDefFile
("/usr/local/tomcat/webapps/axis/WEB-INF/classes/pao/Programers.xml");
```

**ページの開始・終了宣言の方法**

プログラムから帳票にデータをセットする場合は、レポート定義ファイルを読み込んだ後、ページ毎に、ページの開始宣言及びページの終了宣言をしなければなりません。
ページの開始宣言とページの終了宣言の間で帳票データをセットしますが、デザイナー等で作成されたレポート定義ファイルの内容通り帳票を作成するのであれば、データのセットは不要です。
つまりプログラムからレポート定義体を読み込んで帳票データをクライアントに渡す最小構成は、

① 印刷・プレビューインスタンスの生成
② レポート定義ファイルの読込
③ ページの開始宣言
④ ページの終了宣言
⑤ バイト型変数へ印刷データを格納し、クライアントへ復帰。

ということになります。
通常の利用では、「③ページ開始宣言」と「④ページ終了宣言」の間に帳票データをセットするロジックが入る事が殆どでしょう。

ページ開始宣言・ページ終了宣言を行うには、
IReport インタフェースに実装されている pageStart / pageEnd メソッドを使用します。
引数はありません。

＜例＞

```
//頁開始を宣言
paoRep.pageStart();
        ⋮
//頁終了を宣言
paoRep.pageEnd();
```

Write() ･･･印刷データセット処理

**オブジェクトへのデータセット方法**

ここでは、どのようにしてレポート定義ファイルで指定された各オブジェクトに対して値を入れたり、表の横罫線を繰返し描画するのかについて述べていきます。
なお、オブジェクトのデータセットは、必ずページの開始宣言(pageStart)とページの終了宣言(pageEnd)の間で行ってください。

プログラムから帳票にデータをセットする場合は、IReport インタフェースに実装されている write メソッドを使用します。write メソッドは、３つのパターンにオーバーロードされています。

(1) void write(String name, String value)

オブジェクトに対して文字列をセットします。
ヘッダやフッタなど繰り返さない固定オブジェクトの値のセットに使用してください。

String name

レポート定義ファイル内のオブジェクト名を指定します。
この場合、文字列をセットするため対象となるオブジェクトのタイプは、
Text(文字列) と、ArtText(装飾文字列) と、Barcode(バーコード) のみとなります。

String value

セットする文字列を指定します。

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

オブジェクトに対して描画位置を指定して文字列をセットします。

表の行など繰り返し値をセットするオブジェクトに使用してください。

このパターンのメソッドを使用する場合、レポート定義ファイル内の IntervalX 又は IntervalY に１以上の値が入っている必要があります。

IntervalX とは、横方向に繰り返す間隔(mm)です。

IntervalY とは、縦方向に繰り返す間隔(mm)です。主に表の行などに使用されます。

String name

レポート定義ファイル内のオブジェクト名を指定します。

この場合、文字列をセットするため対象となるオブジェクトのタイプは、Text(文字列) と、ArtText(装飾文字列)と、Barcode(バーコード) のみとなります。

String value

セットする文字列を指定します。

int index

IntervalX／IntervalY で指定された縦方向･横方向の間隔で描画を行うページ内の描画位置です。左上から右下方向に値が大きくなります。

例えば、表で IntervalY に値がある場合、描画位置は・・・

オブジェクトの最初の位置＋InterbalY×(index –1) のようになります。

表の場合、１行目が１、２行目が２、３行目が３となります。

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

オブジェクトに対して描画位置を指定します。

表の行の横罫線など繰り返し描画を行うオブジェクトに使用してください。

このパターンのメソッドを使用する場合、レポート定義ファイル内の IntervalX 又は IntervalY に１以上の値が入っている必要があります。

IntervalX とは、横方向に繰り返す間隔(mm)です。

IntervalY とは、縦方向に繰り返す間隔(mm)です。主に表の行などに使用されます。

String name

レポート定義ファイル内のオブジェクト名を指定します。

この場合、どのオブジェクトでも繰返し描画ができるため対象となるオブジェクトのタイプは、全てのオブジェクトです。

int index

IntervalX／IntervalY で指定された縦方向･横方向の間隔で描画を行うページ内の印字位置です。左上から右下方向に値が大きくなります。

例えば、表で IntervalY に値がある場合、印字位置は・・・

オブジェクトの最初の位置＋InterbalY×(index –1) のようになります。

表の場合、１行目が１、２行目が２、３行目が３となります。

＜例＞

```
int page = 0; //頁数を定義
int line = 0; //行数を定義
for (int i = 0; i < 60; i++)
{
        if (i % 15 == 0) //1頁15行で開始
        {
                //頁開始を宣言
                paoRep.PageStart();
                page++;             //頁数をインクリメント
                line = 0;           //行数を初期化

                //＊＊＊ヘッダのセット＊＊＊
                //文字列のセット
                GregorianCalendar cal = new GregorianCalendar();
                paoRep.write("DateTime", cal.getTime().toString());
                paoRep.write("Page", "Page - " + Integer.toString(page));
        }
        line++; //行数をインクリメント

        //＊＊＊明細のセット＊＊＊
        //繰返し文字列のセット
        paoRep.write("LineNo", Integer.toString(i+1) , line);
        paoRep.write("10Baisu", Integer.toString((i+1)*10) , line);
        //繰返し図形(横線)のセット
        paoRep.write("HLine", line);

        if (((i+1) % 15) == 0) paoRep.PageEnd(); //1頁15行で終了宣言
}
```

## 圧縮した印刷バイナリデータ取得

プログラムから圧縮した印刷バイナリデータの取得を行うには、IReport インタフェースに実装されている saveData メソッドを使用します。引数はありません。
戻り値に、byte[] 型の圧縮データが、格納されます。

# プログラマーズリファレンス

## IReport インターフェース

Reports.jar を制御する全てのメソッドを保持しているインターフェースです。
ReportCreator クラスの持つ getReport メソッドによりインスタンスを生成することが可能です。印刷時、getReport にてインスタンスを生成してください。

コンストラクタ
引数なし

パブリックメソッド

| | |
|---|---|
| loadDefFile | レポート定義ファイルを読み込む |
| pageStart | ページの開始を宣言する |
| pageEnd | ページの終了を宣言する |
| write | 印刷データを書き込む |
| saveXMLFile | 印刷データファイルを書き出す |
| saveData | 圧縮した印刷バイナリデータを返す |

### ReportCreator クラス

印刷行うオブジェクトを返すメソッドを実装したクラスです。

IReport 型の getReport メソッドを内蔵しています。

getReport メソッドを呼び出して印刷してください。

パブリックメソッド

| | |
|---|---|
| getReport | 印刷オブジェクトを返す |

## getReport メソッド

印刷を制御するオブジェクトを返すメソッドです。
印刷オブジェクトインスタンスの生成になりますので、最初に必ず行ってください。

＜例＞

IReport GetReport()

```
//印刷オブジェクトのインスタンスを獲得
paoRep = ReportCreator.GetReport();
```

**参照**

ReportCreator クラス

loadDefFile メソッド

レポート定義ファイルを読み込みます。
プログラムがどこで動作するかわからないため、絶対パスを指定することをお勧めします。

＜例＞

paoRep.loadDefFile(String name)

String name

　　レポート定義ファイル名

```
//レポート定義ファイルの読み込み
paoRep.loadDefFile
("/usr/local/tomcat/webapps/axis/WEB-INF/classes/pao/Programers.xml");
```

**参照**

IReport インターフェース

## pageStart メソッド

ページの開始宣言をします。
ページの開始を宣言後、ページの終了宣言(pageEnd)までの間に、印刷データをセットするコードを入れてください。

＜例＞
void pageStart()

```
//頁開始を宣言
paoRep.pageStart();
        ⋮
//頁終了を宣言
paoRep.pageEnd();
```

write() ･･･印刷データセット処理

**参照**

IReport インターフェース

## pageEnd メソッド

ページの終了宣言をします。
ページの開始宣言(pageStart)から、このメソッドの宣言までの間に、印刷データをセットするコードを入れてください。

＜例＞
void pageEnd()

```
//頁開始を宣言
paoRep.pageStart();
        ⋮
//頁終了を宣言
paoRep.pageEnd();
```

write() ･･･印刷データセット処理

**参照**

IReport インターフェース

## write メソッド

レポート定義ファイルで指定されたオブジェクトの操作を行います。
レポート定義ファイルで指定されているオブジェクトに対して文字を書き込んだり、表の横罫線を繰返し描画したりします。

### オーバーロードの一覧

void write(String name, String value)
　オブジェクトに対して文字列をセットします。
　ヘッダやフッタなど繰り返さない固定オブジェクトの値のセットに使用してください。
void write(String name, String value, int index)
　オブジェクトに対して描画位置を指定して文字列をセットします。
　表の行など繰り返し値をセットするオブジェクトに使用してください。
void write(String name, int index)
　オブジェクトに対して描画位置を指定します。
　表の行の横罫線など繰り返し描画を行うオブジェクトに使用してください。

### 参照

IReport インターフェース

### void write(String name, String value) メソッド

オブジェクトに対して文字列をセットします。
ヘッダやフッタなど繰り返さない固定オブジェクトの値のセットに使用してください。

String name

レポート定義ファイル内のオブジェクト名を指定します。
この場合、文字列をセットするため対象となるオブジェクトのタイプは、
Text(文字列) と、ArtText(装飾文字列) と、Barcode(バーコード) のみとなります。

String value

セットする文字列を指定します。

＜例＞

```
//文字列のセット
paoRep.write("DateTime", cal.getTime().toString());
```

**参照**

IReport インターフェース

### void write(String name, String value, int index) メソッド

オブジェクトに対して描画位置を指定して文字列をセットします。
表の行など繰り返し値をセットするオブジェクトに使用してください。
このパターンのメソッドを使用する場合、レポート定義ファイル内の IntervalX 又は IntervalY に１以上の値が入っている必要があります。
IntervalX とは、横方向に繰り返す間隔(mm)です。
IntervalY とは、縦方向に繰り返す間隔(mm)です。主に表の行などに使用されます。

String name

レポート定義ファイル内のオブジェクト名を指定します。
この場合、文字列をセットするため対象となるオブジェクトのタイプは、 Text(文字列) と、ArtText(装飾文字列)と、Barcode(バーコード) のみとなります。

String value

セットする文字列を指定します。

int index

IntervalX／IntervalY で指定された縦方向･横方向の間隔で描画を行うページ内の描画位置です。左上から右下方向に値が大きくなります。
例えば、表で IntervalY に値がある場合、描画位置は・・・
オブジェクトの最初の位置＋InterbalY×(index –1) のようになります。
表の場合、１行目が１、２行目が２、３行目が３となります。

＜例＞

```
//繰返し文字列のセット
paoRep.write("Page", "Page - " + Integer.toString(page));
```

**参照**

IReport インターフェース

### void write(String name, int index) メソッド

オブジェクトに対して描画位置を指定します。
表の行の横罫線など繰り返し描画を行うオブジェクトに使用してください。
このパターンのメソッドを使用する場合、レポート定義ファイル内の IntervalX 又は IntervalY に１以上の値が入っている必要があります。
IntervalX とは、横方向に繰り返す間隔(mm)です。
IntervalY とは、縦方向に繰り返す間隔(mm)です。主に表の行などに使用されます。

String name

レポート定義ファイル内のオブジェクト名を指定します。
この場合、どのオブジェクトでも繰返し描画ができるため対象となるオブジェクトのタイプは、全てのオブジェクトです。

int index

IntervalX／IntervalY で指定された縦方向･横方向の間隔で描画を行うページ内の印字位置です。左上から右下方向に値が大きくなります。
例えば、表で IntervalY に値がある場合、印字位置は・・・
オブジェクトの最初の位置＋InterbalY×(index –1) のようになります。
表の場合、１行目が１、２行目が２、３行目が３となります。

＜例＞

```
line++; //行数をインクリメント

//＊＊＊明細のセット＊＊＊
//繰返し文字列のセット
paoRep.write("LineNo", Integer.toString(i+1) , line);
paoRep.write("10Baisu", Integer.toString((i+1)*10) , line);
//繰返し図形(横線)のセット
paoRep.write("HLine", line);
```

**参照**

IReport インターフェース

### saveData メソッド

圧縮した印刷バイナリデータを返します。
WEB サービス側で、リッチクライアントに返す印刷データを作成する時に使用します。

＜例＞
byte[] saveData()

```
byte[] b = paoRep. saveData(); // 圧縮した印刷バイナリデータを返す
```

**参照**

IReport インターフェース

## saveXMLFile メソッド

印刷データを XML ファイルに保存します。

＜例＞

bool saveXMLFIle(String name)

String name

保存する印刷データ XML ファイルパス名

```
paoRep.saveXMLFile("印刷データ.XML"); //印刷データの保存
```

**参照**

IReport インターフェース

## 変更履歴

| 版 | 作成日 | 変更点 |
|---|---|---|
| 1 | 2006.08.01 | 新規作成 |