

2004.12.03 バージョン 0.4.0 に関する記述を追加しました。
それ以外については、WEB+DB PRESS Vol.17 に寄稿した記事の方がわかりやすいと思うので、初めての方は WEB+DB PRESS の記事を読んだ後、『4. 追記 バージョン 0.4.0 までの変更点』を読んでもらうのが良いかと思います。
http://www.bbreak.co.jp/technique/doc/ireport/iReport_webdb.pdf

1. 基本説明	2
1.1. iReport とは	2
1.2. 目的	2
1.3. インストール	3
1.4. 事前準備	4
1.5. 帳票作成	6
1.6. コンパイル	13
1.7. 実行	14
2. Web システムへの組み込み	15
3. 開発 Tips	19
3.1. クエリに任意のパラメータを使用する方法	19
3.2. 計算結果を表示する方法	20
3.3. コンパイルエラーの原因を調査する方法	22
4. 追記 Version0.4.0 までの変更点	23
4.1. 日本語対応	23
4.2. メイン画面	24
4.3. データソース	25
4.4. その他	26
5. 参考文献	27
6. 最後に	28

bBreak

1. 基本説明

1.1. iReport とは . . .

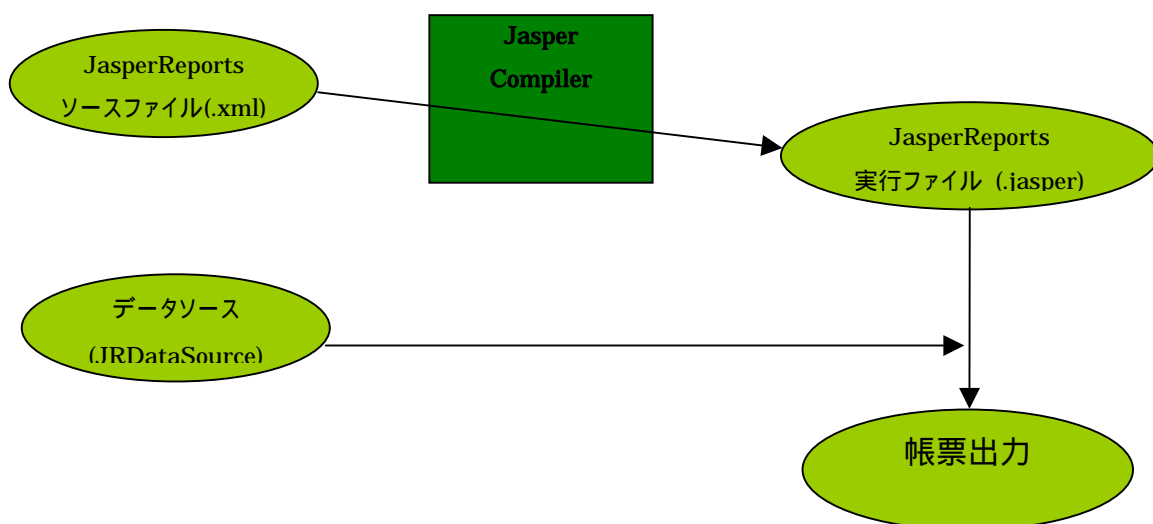
iReport とは JasperReports のデザインツールで、帳票フォーマットの編集から実行までを GUI から行なうためのオープンソースのツールです。

JasperReports とは Java で帳票出力を実現するオープンソースのライブラリで、同一のテンプレートから PDF,HTML,Excel 等に帳票を出力することができます。

iReport、JasperReports それぞれの詳細については以下の URL を参照してください。

iReport: <http://ireport.sourceforge.net/>

JasperReports: <http://jasperreports.sourceforge.net/>



1.2. 目的

iReport、JasperReports のインストールから帳票作成、データベース情報を使った動的データバインドまでの方法について簡単に説明します。

また、2Web システムへの組み込みでは作成した帳票を Web システムに組み込む方法について説明します。

1.3. インストール

今回使用した動作環境を以下に示します。

クライアント動作環境

OS: Windows2000 Professional
JavaVM: Sun JDK 1.4.1
ant: 1.4
iReport: iReport 0.2.1
JasperReports: JasperReports 0.5.0

サーバ動作環境

OS: Red Hat Linux 8.0
DB: PostgreSQL 7.2.1

(1) ダウンロード

iReport (iReport-0.2.1.zip) および JasperReports (jasperreports-0.5.0-project.zip) をそれぞれ以下のサイトからダウンロードします。

iReport: <http://prdownloads.sourceforge.net/ireport/iReport-0.2.1.zip?download>

JasperReports: <http://prdownloads.sourceforge.net/jasperreports/jasperreports-0.5.0-project.zip?download>

(2) インストール

iReport、JasperReports をそれぞれクライアントの任意のディレクトリに解凍します。

ここではそれぞれ C ドライブ直下に解凍したものとします。(C:¥iReport-0.2.1、C:¥jasperreports-0.5.0)

iReport の起動ファイル (C:¥iReport-0.2.1¥iReport.bat) を環境に合わせて設定します。

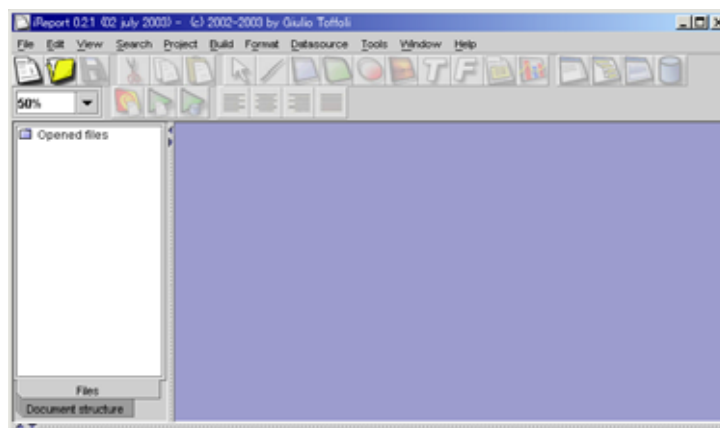
```
@echo off

set JAVA_HOME=C:¥j2sdk1.4.0_03
set ANT_HOME=C:¥ant
set IREPORT_HOME=C:¥Documenti¥progetti¥iReport¥iReport2¥

rem %ANT_HOME%¥bin¥ant javadocs
%ANT_HOME%¥bin¥ant iReport
```

bBreak

iReport の起動ファイル(C:\¥iReport-0.2.1¥iReport.bat)を実行すると iReport が起動します。



1.4. 事前準備

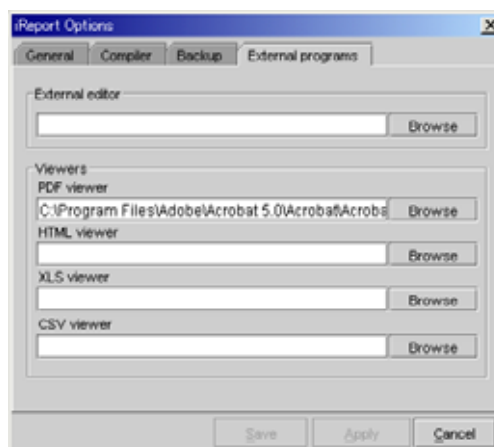
帳票を作成する前に、iReport に必要な設定を行ないます。

(1) 外部プログラム設定

帳票出力時に、表示に使用するプログラムを設定します。

メニューで[Tools]-[Options]を開き[External programs]を選択します。

今回は pdf での表示を確認するので[PDF Viewer]を指定します。



(2) データソース設定

データベースから取得した情報を帳票に組み込む際に必要となる設定です。

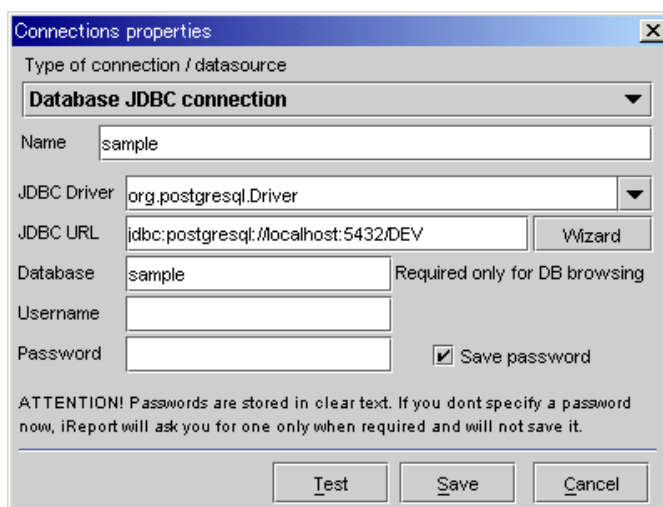
(A) データソース生成

メニューで[DataSource]-[Connections / Datasources]を選択します。

[New]を選択し、データベースへの接続情報を入力します。

使用する JDBC ドライバは予め iReport の lib (C:\¥iReport-0.2.1¥lib) にコピーし、iReport を再起動しておきます

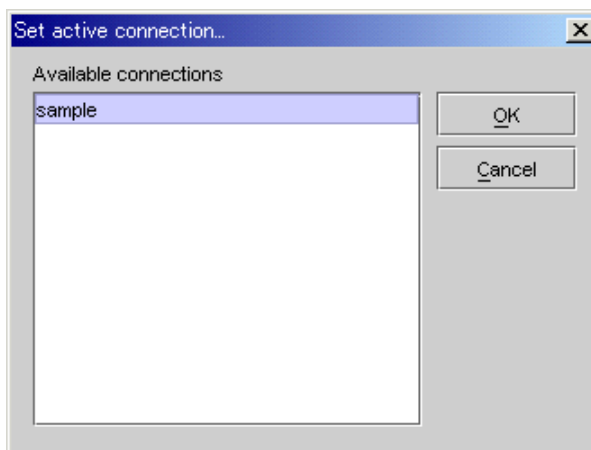
[Test]を押して、データベースへの接続が確認できたら、[SAVE]して終了します。



(B) データソース設定

メニューで[Build]-[Set Active connection]を選択します。

先ほど作成したデータソースを選択し、[OK]を押します。



1.5. 帳票作成

帳票のフォーマット作成を行ないます。

メニューで[File]-[New document]を選択します。

[Report name]を入力し、その他はそのまま[OK]を押します。

Report properties

Report name:

Page size

Preset sizes:

Width: cm

Height: cm

Orientation:

Page Margin Columns Scriptlet More...

Page margin

Top: cm

Bottom: cm

Left: cm

Right: cm

OK

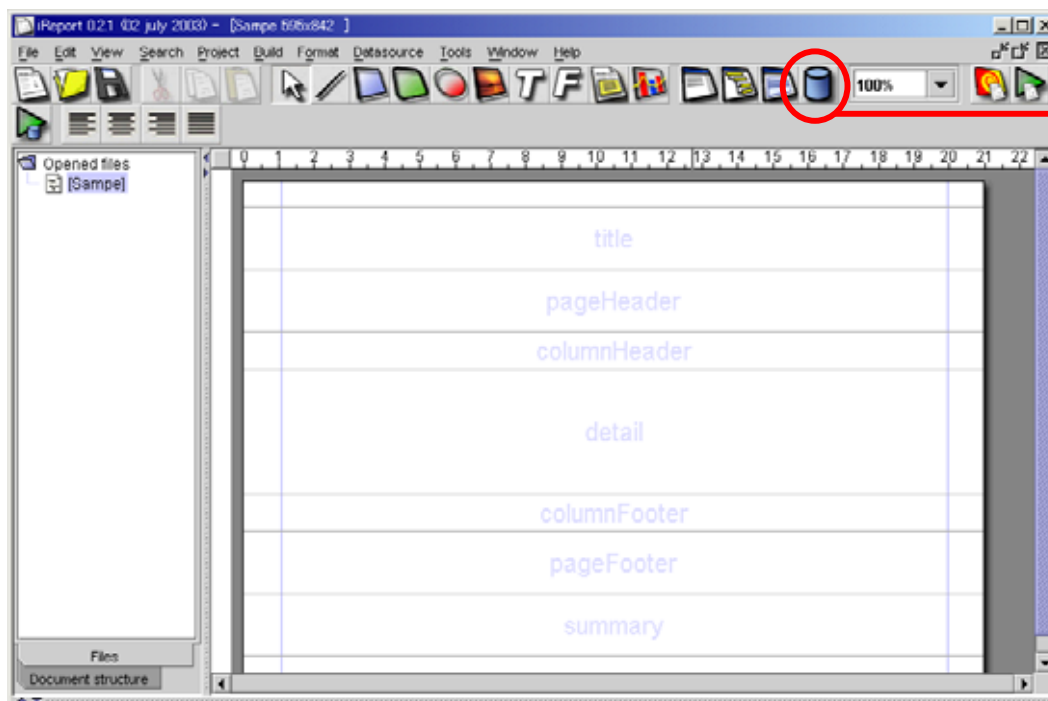
Cancel

bBreak

(1) クエリー作成

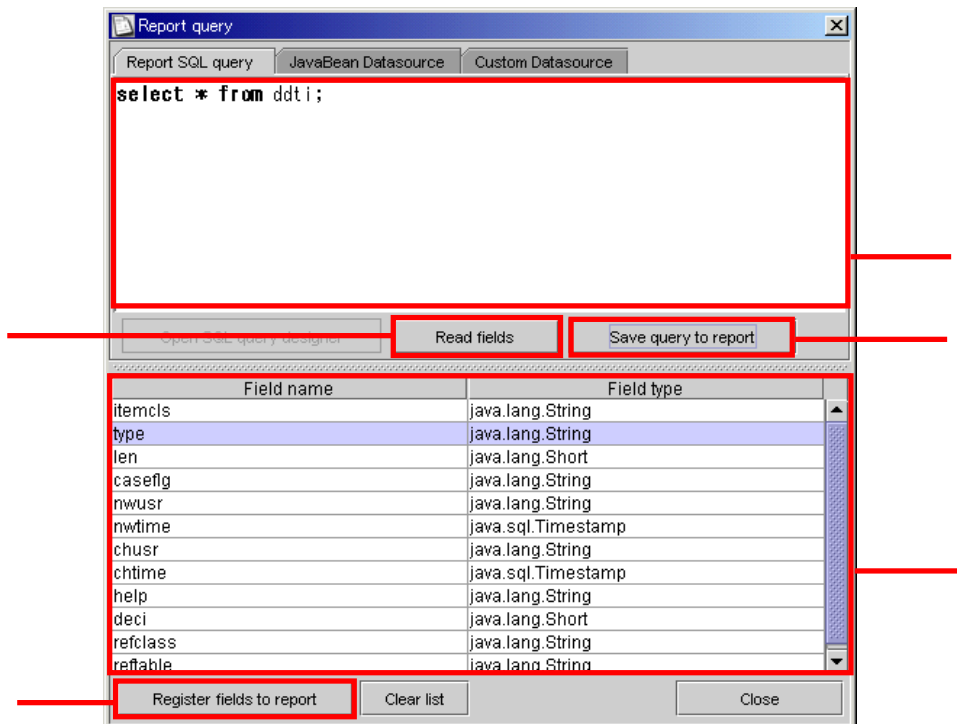
表示内容をデータベースから取得するためのクエリーを作成します。

下図中 (もしくはメニューから[Datasource]-[Report query])を選択します。



bBreak

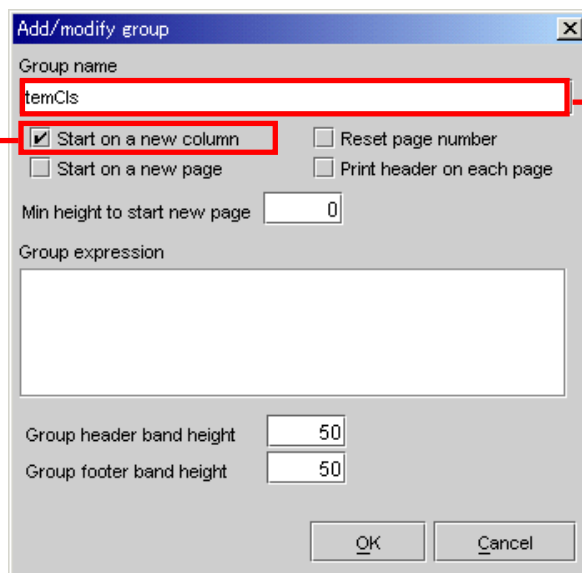
[Report SQL Query]を選択し、表示フィールドを取得するSQLを記述します(下図中)
SQL入力後[Read fields(下図中)]を押すと、フィールド一覧が表示されます(下図中)
一覧(下図中)から表示に使用するフィールドを選択し、[Register fields to report(下図中)]を選択します。
最後に[Save query to report(下図中)]を押し、クエリーを保存し終了します。



bBreak

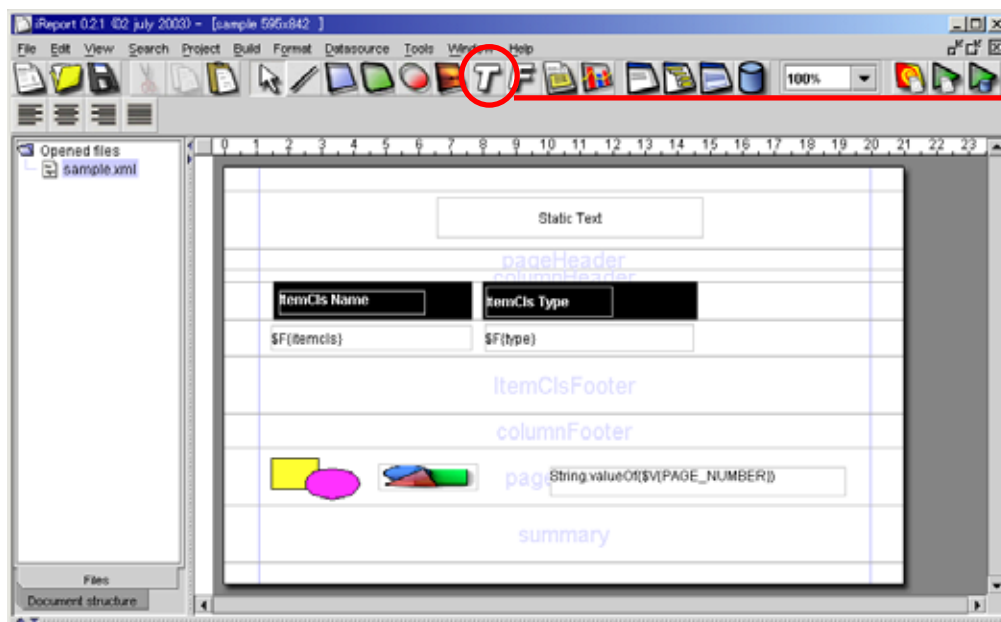
(2) グループ作成

グループという単位で表示部をまとめることで、ページ単位だけでなく、任意の範囲にヘッダー、フッターを設定することが可能となります。



(3) 静的な文字列の定義

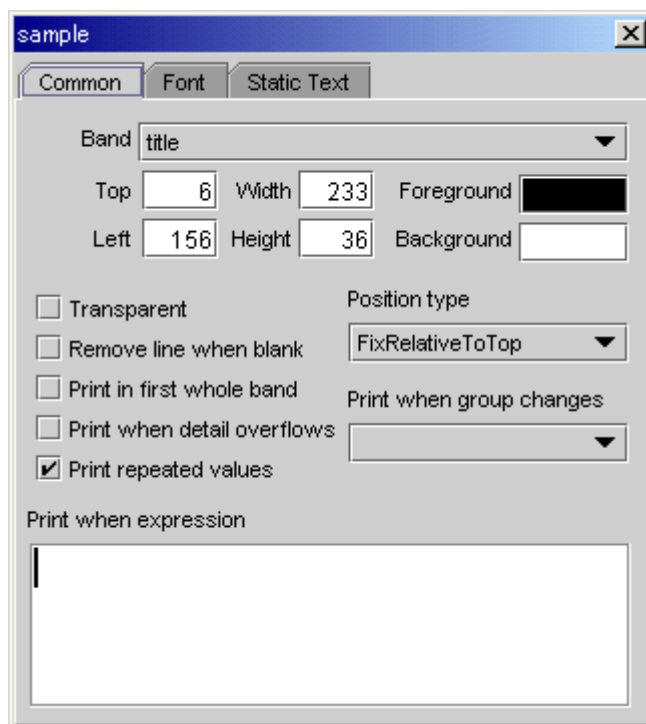
下図中 を選択し、任意の場所に配置します。



bBreak

配置したコンポーネントをダブルクリックすると、以下のようなプロパティウィンドウが表示され、各種プロパティを編集できます。


[Band]の項目はこのコンポーネントが帳票上のどのフィールドに属するかを指定します。




今回の例では[title]および[itemClsHeader]にテキストを追加しています。

(4) フィールド定義

取得したテーブル情報を出力するためにフィールド定義を行ないます。

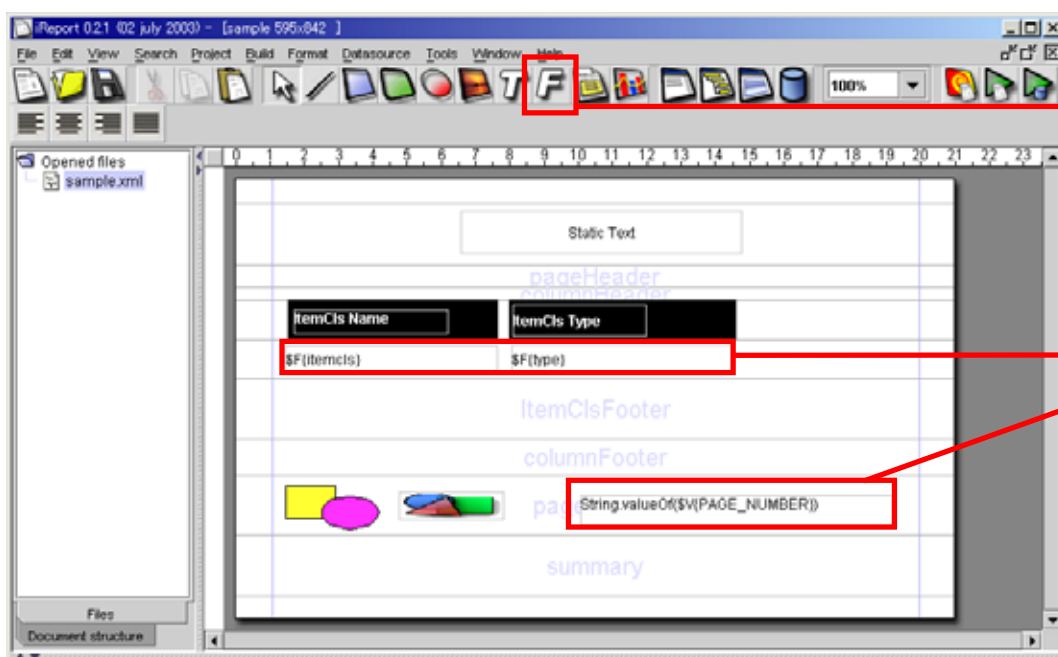
フィールド定義を作成するには下図中  を押し、任意の場所にフィールド定義を配置します。

(例) SF{itemcls}

 を押して各タグに出力される情報が使用可能です。

修飾子	説明
SF	フィールドを出力する際に使用する修飾子。 [Fields]タグに表示されている内容が指定可能。
SV	変数を出力する際に使用する修飾子。 [Variables]タグに表示されている内容が指定可能。
SP	パラメータを出力する際に使用する修飾子。 [Parameter]タグに表示されている内容が指定可能。

今回は[detail]上にSF{itemcls},SF{type}, pageFooter に String.valueOf(SV{PAGE_NUMBER})を配置しています。



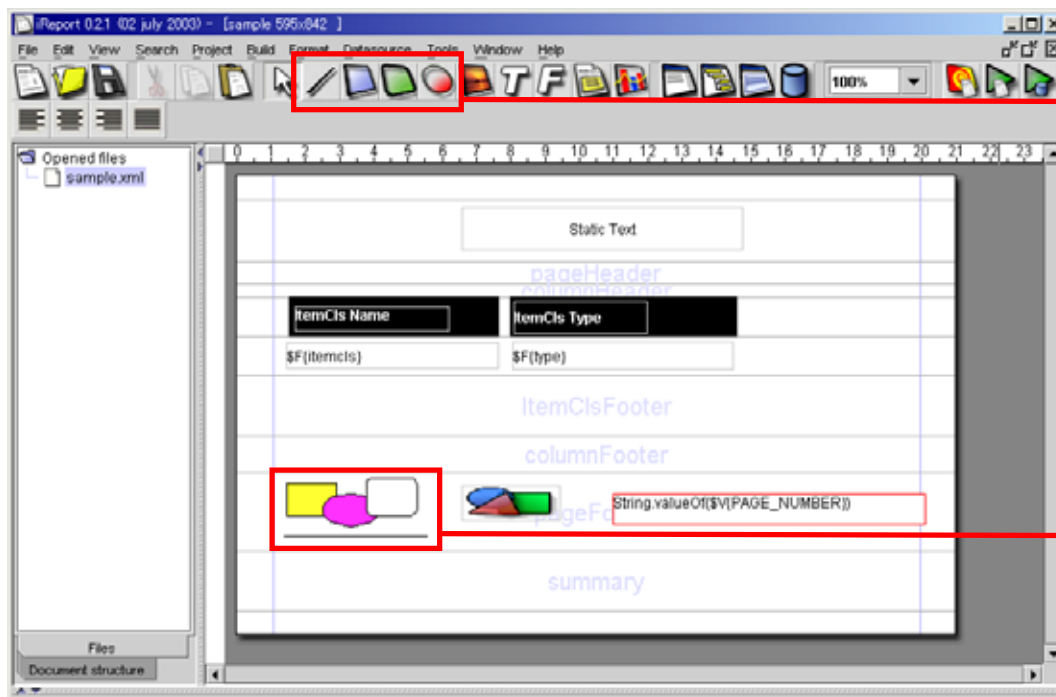
bBreak

(5) 図形作成

サンプルということで、図形も表示してみましょう。

図形コンポーネント[下図中]を選択し、帳票上の任意の場所に配置します。

今回は pageFooter(下図中)に配置しています。



(6) 画像作成


続いて画像も配置してみましょう。と行きたいところなのですが、xml ファイル保存時にファイルパスが消えてしまい、画像の出力が正常に行なえません。


iReport から xml ファイルを保存後、手動で image にパス情報を追加すると画像を正常に表示することが可能です。
(iReport 上からは実行できません)

表示方法詳細は、2(4)(D)画像の表示を参照してください。

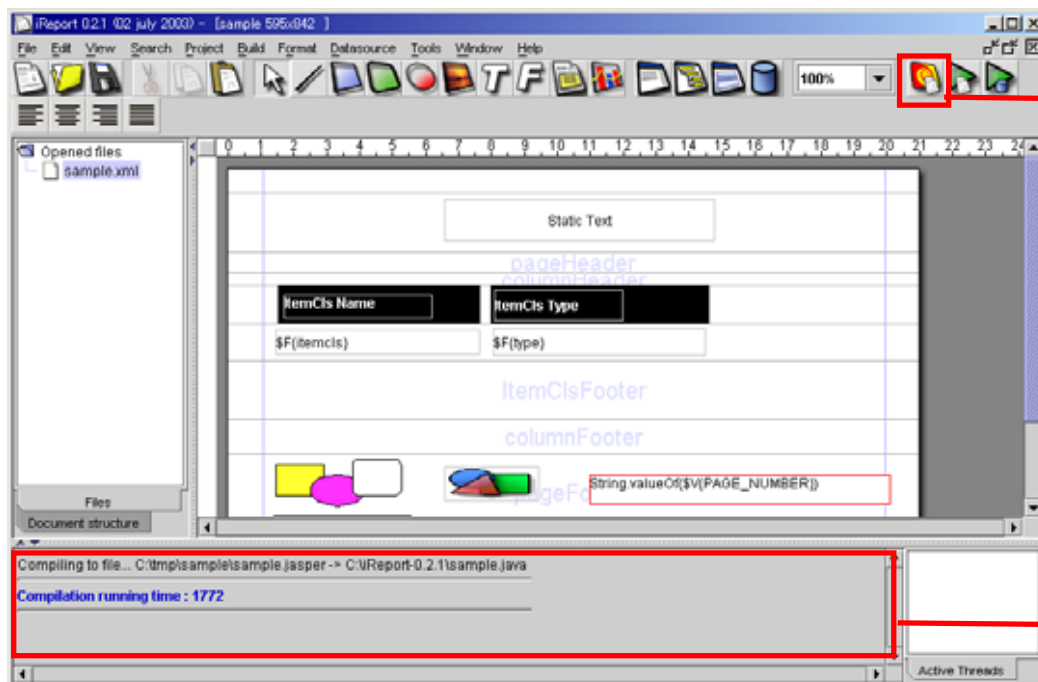
bBreak

1.6. コンパイル

実行に先立ち、まずは下図中  を押して、コンパイルを行い定義に異常が無いことを確認します。

コンパイルが異常終了した場合は下図中  に赤字でエラーメッセージが表示されます。

以下のように青字でコンパイル時間が表示されていればコンパイルは成功です。



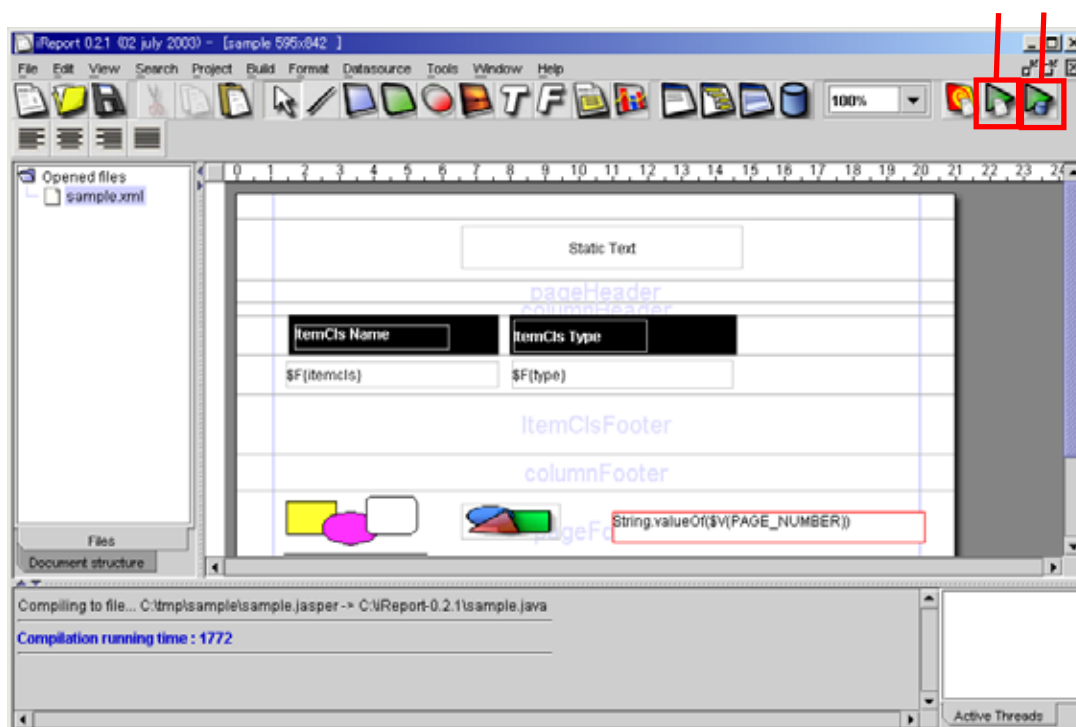
bBreak

1.7. 実行

実行には、コネクションを使用せず、テーブルからのデータ取得を行わない方法(下図中)と、設定したコネクションを使用してデータ取得を行なう方法(下図中)の2通りがあります。

それぞれのパターンで実行し、結果を比較してみましょう。

また、pdf 以外での出力を確認したい場合は、メニューから[Build]-[*** View]で任意の出力形式を選択し、1.4(1)外部プログラム設定と同様に起動するプログラムを指定してから実行します。



2. Web システムへの組み込み

ローカルでの帳票出力が正常に行なわれたところで、ここで作った帳票を Web システムに組み込みます。

なお、TOMCAT インストールディレクトリをここでは CATALINA_HOME とし、JasperReports のインストールディレクトリを JASPER_HOME とします。

サーバ動作環境
OS: Red Hat Linux 8.0
JavaVM: Sun JDK 1.4.1
DB: PostgreSQL 7.2.1
ant: 1.4
tomcat: 4.1.24
JasperReports: JasperReports 0.5.0

(1) サンプルアプリケーションのコピー

\$ CATALINA_HOME/webapps 配下に \$JASPER_HOME/demo/samples/webapps をコピーします。

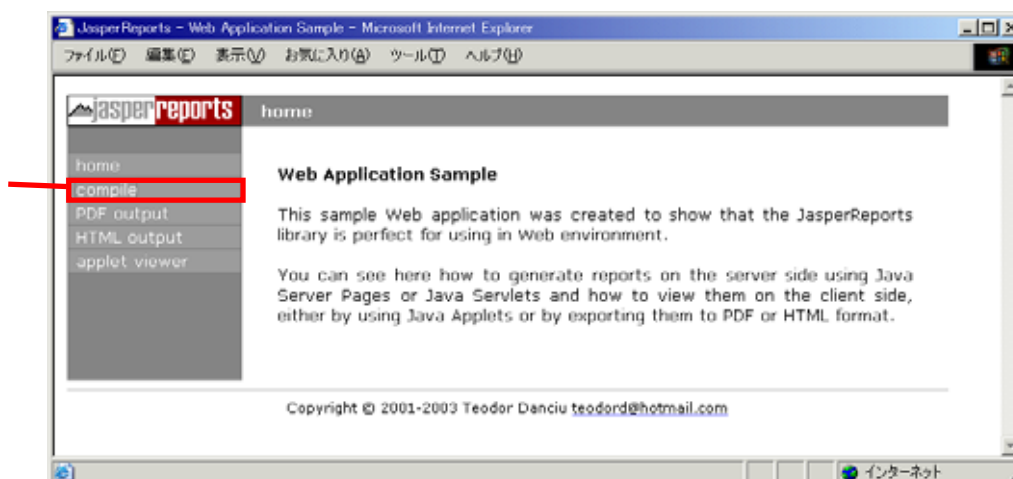
コピーしたディレクトリの名称を JasperReport に変更し、TOMCAT を起動します。

(2) サンプルアプリケーションのコンパイル

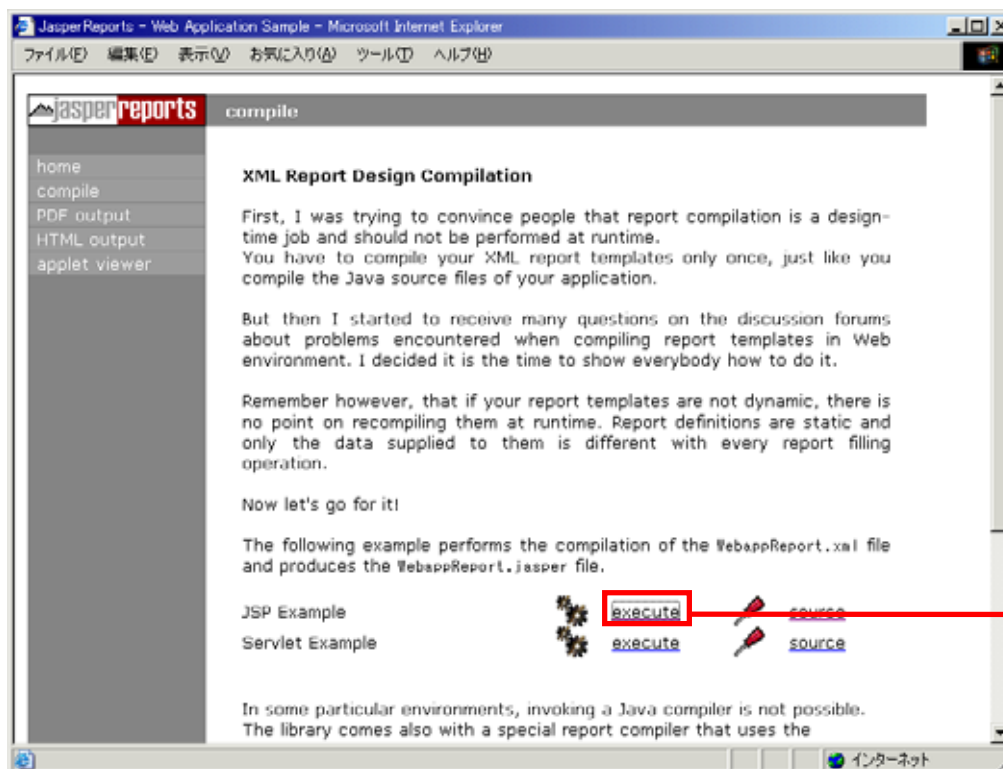
TOMCAT 起動後、以下のような URL にアクセスします。

[http://ホスト名\(IP アドレス\):ポート番号/JasperReport](http://ホスト名(IP アドレス):ポート番号/JasperReport)

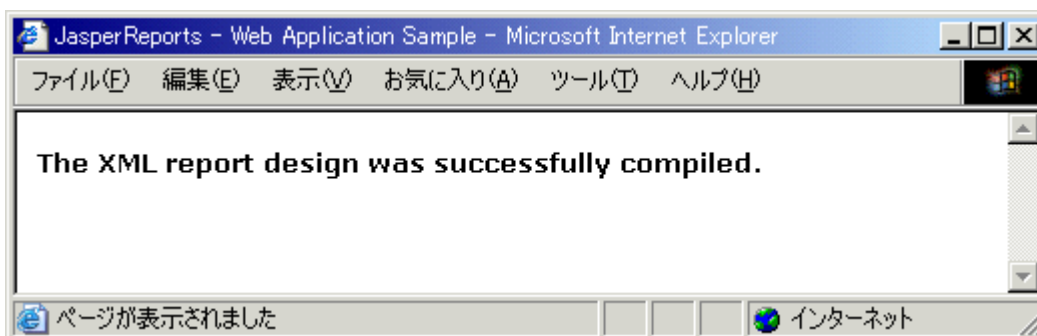
compile(下図中)を選択し、コンパイル画面を開きます。



execute(下図中)を押し、コンパイルを実行します。



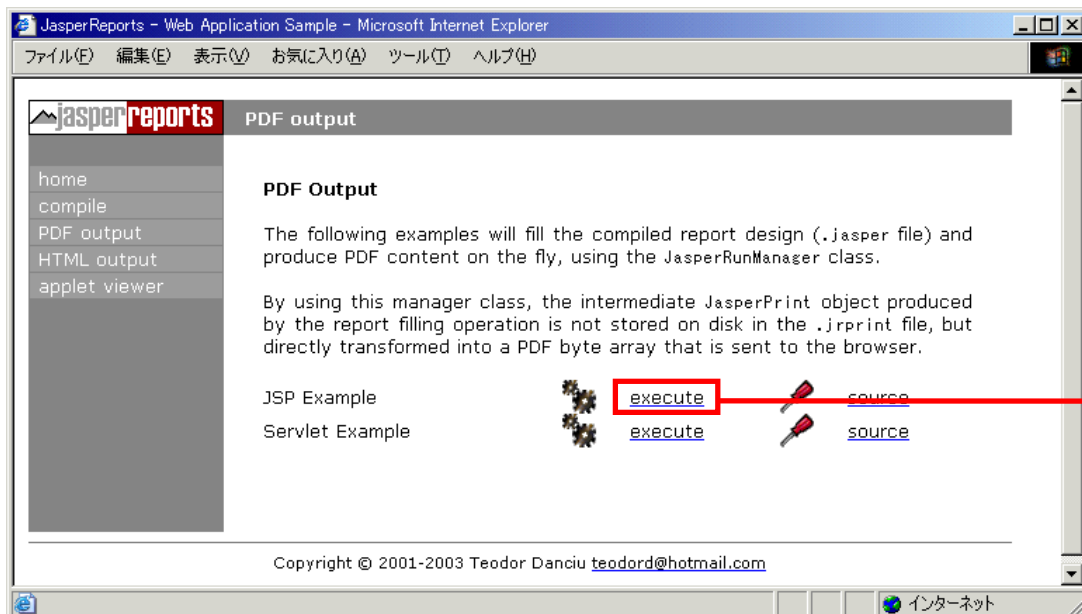
以下のように表示されればコンパイルは成功です。



(3) サンプルアプリケーションの実行

home に戻り[PDF output(下図中)]を押し、execute(下図中)を押します。

設定が正常にされているとサンプルの PDF ファイルが出力されます。



(4) 作成した帳票の移行

(A) 作成物のコピー

サンプルアプリケーションが正常に実行できたら、以下のディレクトリに作成物(sample.xml, sample.jasper)をコピーします。

\$ CATARINA_HOME/webapps/JasperReport/reports

(B) JSP の書き換え

SCATARINA_HOME/webapps/JasperReport/JSP の compile.jsp を以下(太字網掛け)のように書き換えます。

```
:  
JasperCompileManager.compileReportToFile(application.getRealPath("/reports/sample.xml"));  
:
```

SCATARINA_HOME/webapps/JasperReport/JSP の pdf.jsp を以下(太字網掛け)のように書き換えます。

```
:  
File reportFile = new File(application.getRealPath("/reports/sample.jasper"));  
:  
// データベースからデータを取得する場合には、JasperRunManager に connection を渡す。  
Class.forName("org.postgresql.Driver");  
Connection con = DriverManager.getConnection("jdbc:postgresql://localhost:5432/DEV","user", "pass");  
  
byte[] bytes =  
    JasperRunManager.runReportToPdf(  
        reportFile.getPath(),  
        parameters,  
        con  
    );  
:
```

(C) 日本語の表示

日本語を表示するためには、予めフォントセットをサーバの任意の場所に配置し、以下のように sample.xml の定義を変更します。

例)MSGothic を使用した場合

```
<font fontName="MSGothic" pdfFontName="/usr/local/fonts/msgothic.ttc,1" size="12" isBold="false"  
isItalic="false" isUnderline="false" isPdfEmbedded = "false" pdfEncoding = "Identity-H"  
isStrikeThrough="false" />
```

(D) 画像の表示

iReport で保存すると消えてしまったファイルパスですが、直接 sample.xml を編集すると表示することができます。

```
<image>  
  <imageExpression class="java.lang.String">  
    <![CDATA["/usr/local/tomcat/jakarta-tomcat-4.1.24/webapps/JasperReports/reports/logo.jpg"]]>  
  </imageExpression>  
</image>
```

3. 開発 Tips

3.1. クエリに任意のパラメータを使用する方法

データベース操作時にプログラムから任意のパラメータを渡してクエリを発行する方法について説明します。

(1) iReport 上もしくは xml ファイルを直接編集し queryString に以下の様に記述します。

SP{任意のパラメータ名}

例)

```
<queryString><![CDATA[select * from dpmn where menuact=SP{AC_CODE};]]></queryString>
```

(2) コンパイル時に型情報は以下の様にパラメータ名、型情報を設定する必要があります。

```
// パス設定
File sourceFile = new File( "paramTest.xml");
JasperDesign jasperDesign = JRXmlLoader.load( absolutePath);
File destFile = new File( absolutePath.substring(0, absolutePath.length() - 3) + "jasper");
String destFileName = destFile.toString();

// 任意のパラメータ設定
JRDesignParameter designParameter = new JRDesignParameter();
designParameter.setName( "AC_CODE");
designParameter.setValueClass( new String().getClass());

jasperDesign.addParameter( designParameter);

JasperCompileManager.compileReportToFile(jasperDesign, destFileName);
```

(3) 帳票出力時にパラメータの値を設定します。

```
// Map parameters = new HashMap();
paramMap.put("BaseDir", "D:¥¥work¥¥");
paramMap.put("AC_CODE", "NEW");

byte[] bytes;
try {
    bytes =
        JasperRunManager.runReportToPdf(
            reportFile.getPath(),
            paramMap,
            con);
}
```

3.2. 計算結果を表示する方法

以下のような構成の帳票を例にとって説明します。

- 小計には単価 × 個数を表示
- 合計には小計の合計を表示

商品名	単価	個数	小計
商品 A	1000	2	2000
商品 B	10000	1	10000
合計			12000

各フィールド名が以下の様に定義されているとします。

単価	\$F{item_qty}
個数	\$F{item_amount}

以下の様に変数名(下図中)、型情報(下図中)、計算式(下図中)を入力し、小計の変数を作成します。

The dialog box 'Add/modify variable' contains the following fields and values:

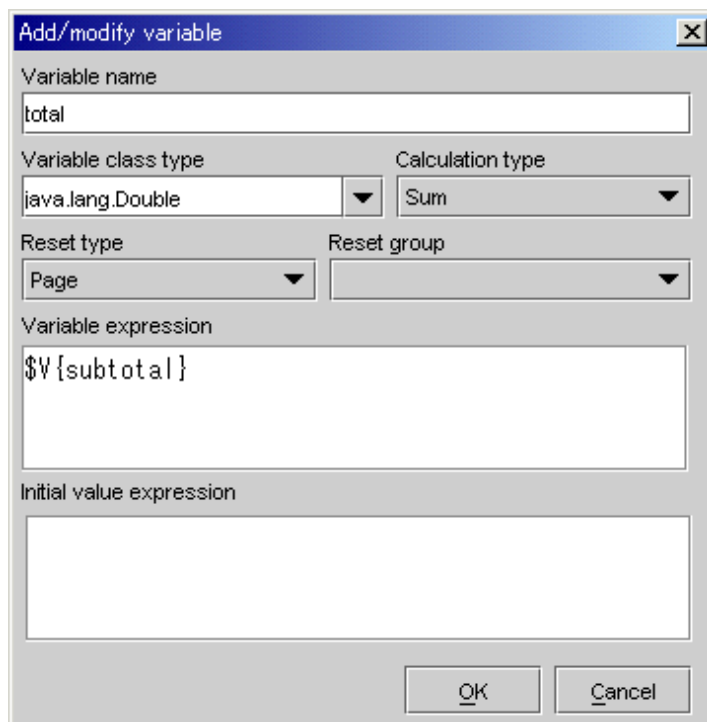
- Variable name: subtotal
- Variable class type: java.lang.Double
- Calculation type: Nothing
- Reset type: None
- Reset group: (empty)
- Variable expression: `new Double($F{item_qty}.doubleValue() * $F{item_amount}.doubleValue())`
- Initial value expression: (empty)

bBreak

同様に合計の変数も定義します。

以下の様に定義すると、Reset type (Page)内に存在する Variable expression ($\$V\{subtotal\}$)を Calculation type (Sum)に従って計算した結果が $\$V\{total\}$ となります。

今回の例では小計 ($\$V\{subtotal\}$)の足された結果が合計 ($\$V\{total\}$)となります。



変数を定義したら、フィールドを表示する場合と同様に、Textfield expression を $\$V\{subtotal\}$ および $\$V\{total\}$ として配置します。

以下に定義例と実行例を示します。

$\$F\{item_qty\}$	$\$F\{item_amount\}$	$\$V\{subtotal\}$
	Total	$\$V\{total\}$
1.0	1.0	1.0
2.0	3.0	6.0
3.0	4.0	12.0
5.0	7.0	35.0
6.0	80000.0	480000.0
Total		480054.0

3.3. コンパイルエラーの原因を調査する方法

iReport 上で帳票を作成している際に何らかのエラーによってコンパイルが通らなくなってしまう場合があります。

iReport 上のメッセージではどの部分が悪いのか分からないような場合に以下の方法を使用すると詳細な情報を得られます。

(1) iReport 上でコンパイル

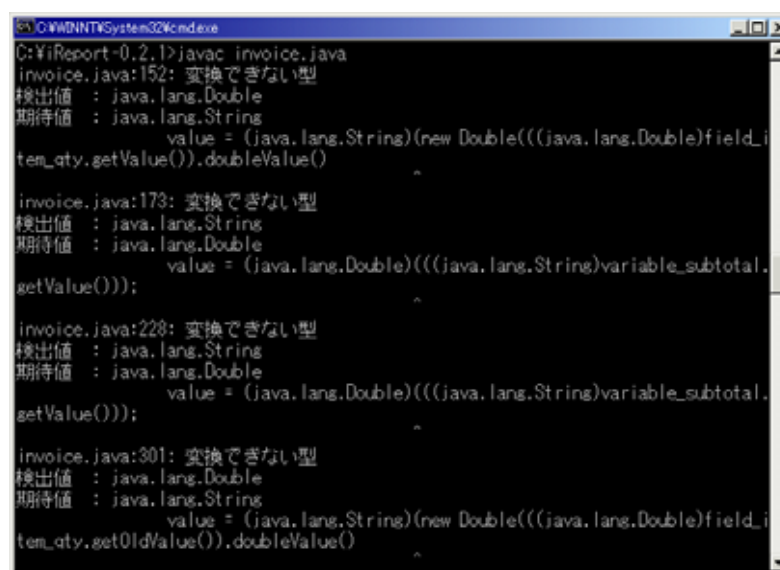
エラーが出るがどこが悪いのか良くわからない。

```
Compiling to file... C:\tmp\demo\invoice.jasper-> C:\iReport-0.2.1\invoice.java
Errors compiling C:\tmp\demo\invoice.jasper!

java.io.FileNotFoundException: C:\iReport-0.2.1\invoice.class (XXXXXXXXXXXXXXXXXXXX)
  at java.io.FileInputStream.open(Native Method)
  at java.io.FileInputStream.<init>(FileInputStream.java:103)
  at dori.jasper.engine.design.JRCompiler.readClassBytecodes(JRCompiler.java:268)
  at dori.jasper.engine.design.JRCompiler.compileReport(JRCompiler.java:152)
  at dori.jasper.engine.JasperCompileManager.compileReportToFile(JasperCompileManager.java:135)
  at dori.jasper.engine.JasperCompileManager.compileReportToFile(JasperCompileManager.java:123)
  at it.businesslogic.ireport.IReportCompiler.run(IReportCompiler.java:139)
  at java.lang.Thread.run(Thread.java:536)
NESTED BY :
```

(2) コマンドラインから該当する java ソースファイルをコンパイルする。

以下のようなエラーメッセージが出力され、型の定義がおかしいことがわかる。



```
C:\iReport-0.2.1>javac invoice.java
invoice.java:152: 変換できない型
抽出値 : java.lang.Double
期待値 : java.lang.String
    value = (java.lang.String)(new Double(((java.lang.Double)field_i
tem_qty.getValue()).doubleValue())
    ^
invoice.java:173: 変換できない型
抽出値 : java.lang.String
期待値 : java.lang.Double
    value = (java.lang.Double)(((java.lang.String)variable_subtotal.
getValue());
    ^
invoice.java:228: 変換できない型
抽出値 : java.lang.String
期待値 : java.lang.Double
    value = (java.lang.Double)(((java.lang.String)variable_subtotal.
getValue());
    ^
invoice.java:301: 変換できない型
抽出値 : java.lang.Double
期待値 : java.lang.String
    value = (java.lang.String)(new Double(((java.lang.Double)field_i
tem_qty.get0IdValue()).doubleValue())
    ^
```

(3) ClassNotFoundException(com.sun.tools.javac.Main)

グラフを使用する際などに、`java.lang.ClassNotFoundException: com.sun.tools.javac.Main` のようなエラーが出る場合があります。`com.sun.tools.javac.Main` は `tools.jar` に含まれていますので、クラスパスに `$JAVA_HOME/lib/tools.jar` を追加すれば上記エラーは出なくなります。

4. 追記 Version0.4.0 までの変更点

はじめに iReport の紹介を書いてから一年ちょっと、バージョンが 0.2.1 から 0.4.0 まで上がりましたが、未だに本資料のヒット数が高く、多くの方に見ていただけているようなので、Version0.4.0 に関する記述を追加することにしました。

多くのバグフィックスや、気になる追加機能も多いですが、目を引くのは操作性の飛躍的な向上です。

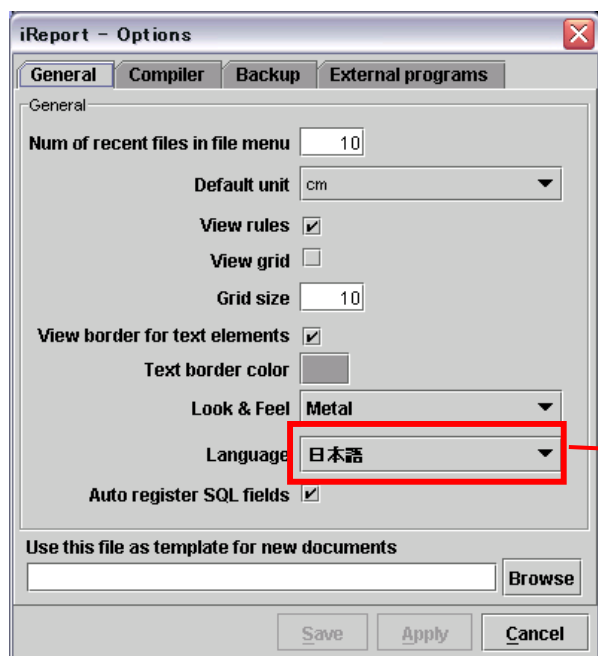
以前はだましまし使用するというイメージだったものが、これなら十分実用に耐えうる完成度になったと思います。以降では主に変更点に絞って説明します。

主な追加機能

- JFreeChart 0.9.21 サポート(Release 0.4.0)
- JasperReport 0.6.1 サポート(Release 0.4.0)
- ツールバーの追加(Release 0.4.0)
- データソース追加 (CSV, XML, Extende Bean) (Release 0.3.2)
- 日本語対応(Release 0.3.2)
- プラグインサポート(Release 0.3.1)
- バーコード対応(実験) (Release 0.3.0)

4.1. 日本語対応

まずは日本語対応を見てみましょう。メニューから [Tools]-[Options...] の [General] タグを開き [Language(下図中)] で日本語を選択し、[Save] を押します。

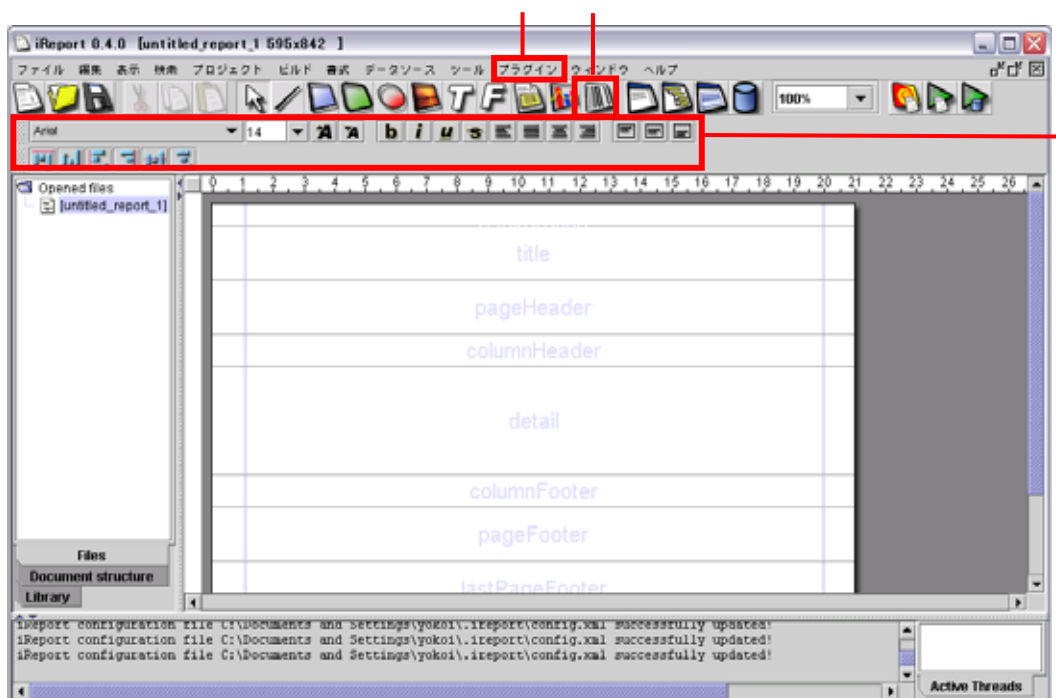


メニューが日本語に変わりましたね。以降は日本語メニューを前提に説明を行います。

bBreak

4.2. メイン画面

ではメニューとツールバーの変更点を見てみましょう。



	プラグイン	プラグインとして追加した機能呼び出す
	バーコード ツール	バーコード設定ツール
	ツールバー	新たに追加されたツールバー

(1) プラグイン

バージョン0.3.1から追加された機能で、`it.businesslogic.ireport.plugin.IReportPlugin`を継承したクラスを作り定義を追加することで、独自の機能を追加できるようになったようです。

標準で用意されている Massive compiler は旧バージョンの定義ファイルを、最新バージョンにコンパイルしてくれる?ようです。

(2) バーコードツール

0.3.0 からバーコードが設定できるようになっています。JAN などの代表的な種類は選択できるようなので、バーコードリーダがあれば試してみたいところです。実験となっているので注意が必要かもしれません。



(3) 新ツールバー

配置したコンポーネントの整列や、テキストの調整など。使い方は直感的に分かると思います。

4.3. データソース

新たに追加になったデータソースを利用してみます。

(1) CSV データソース

CSV と書かれているのですが、カンマ区切りでは上手くいきません。

ソースを見てみると JRCSVDataSource 29 行目

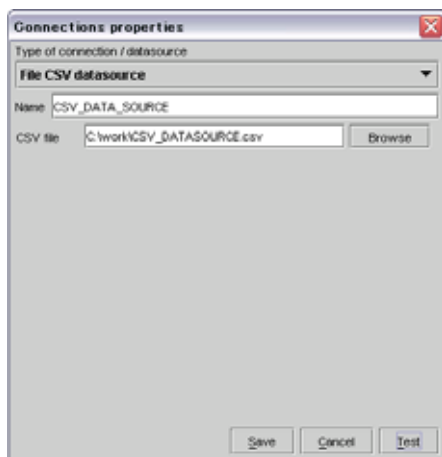
```
StringTokenizer st = new StringTokenizer(row, ",");  
“,”で文字列を分割している！！？
```

というわけで、CSV ファイルを使用したい場合はカスタムデータソースを使用したほうがよさそうです。

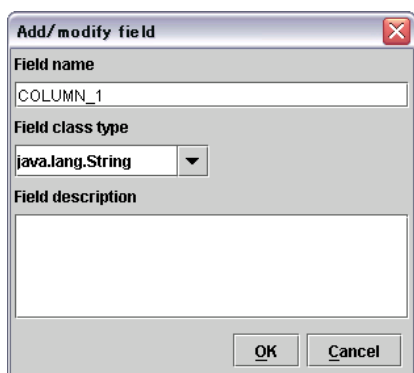
ここでは、以下のようにセミicolon区切りのファイルを用意しておきましょう。

```
1;Software Desing;970;10  
2;WEB+DB Press;1580;15
```

1. CSV のデータソースを定義し、作成したセミicolon区切りのファイルを用意します。



2. セミicolonで区切った値は COLUMN_1, COLUMN_2, COLUMN_3...といったようにアクセスしますので、以下のように Field を定義します。



3. デザイン編集エリアには通常のフィールド定義と同様に以下のように配置します。



4.4. その他

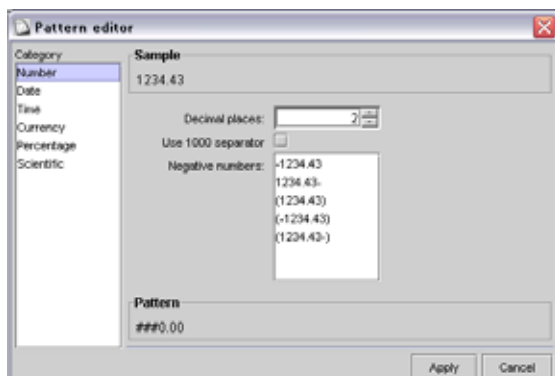
(1) テキストの変更

F2 を押すとテキストの編集ボックスが開きます。



(2) パターンエディタ

テキストの表示形式設定が簡単に出来るようになっています。



5. 参考文献

- WEB+DB PRESS VOL15 / 技術評論社
- iReport 開発ページ(<http://ireport.sourceforge.net/>)
- JasperReports 開発ページ(<http://jasperreports.sourceforge.net/>)

6. 最後に

ドキュメントが不足していたため試行錯誤で苦労しましたが、慣れると楽に帳票が作成でき、特に複数のパターンで出力が求められる場合などは重宝しそうです。

バグらしき点も見受けられますが、そこは今後のバージョンアップに期待といったところでしょう。

ここで紹介した以外にも様々な機能、帳票作成方法がありますが、JasperReports に含まれるサンプルを iReport に読み込ませることで、使用例として参考にすることができます。

記述内容に何かお気づきの点、質問等ありましたら下記までご連絡ください。

開発部 横井 朗 yokoi@bbreak.co.jp
