関係者外秘

PJ関係者限り

[プロジェクト名]

帳票アプリケーション開発ガイド

第１．０版

ＹＹＹＹ年Ｍ月Ｄ日

[会社名]

[部門名]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No. | 版数 | 変更日 | 区分 | 変更箇所 | | 変更内容 | 担当者 |
| 頁 | 項番 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

[1． 本書について 1](#_Toc408304999)

[2． 開発フロー 2](#_Toc408305000)

[2.1. 帳票テンプレートとプログラム 3](#_Toc408305001)

[3． Jaspersoft Studioのインストール 4](#_Toc408305002)

[3.1. Jaspersoft Studioのインストール 4](#_Toc408305003)

[4． 帳票テンプレートの作成 5](#_Toc408305004)

[4.1. 帳票テンプレートのファイル名と格納場所 6](#_Toc408305005)

[4.2. 帳票テンプレートの新規作成 7](#_Toc408305006)

[4.3. 各ビューとバンドの概念 8](#_Toc408305007)

[4.3.1. ビューの説明 8](#_Toc408305008)

[4.3.2. バンドについて 9](#_Toc408305009)

[4.4. 不要なバンドの削除 9](#_Toc408305010)

[4.5. 要素の配置と操作 10](#_Toc408305011)

[4.5.1. タイトルの設定 10](#_Toc408305012)

[4.5.2. パラメータの設定 12](#_Toc408305013)

[4.5.3. 実行ユーザーの見出し要素と値要素の配置 13](#_Toc408305014)

[4.5.4. 出力年月日の設定 15](#_Toc408305015)

[4.5.5. 繰返しデータの設定 16](#_Toc408305016)

[4.5.6. ページフッターの設定 19](#_Toc408305017)

[4.6. 帳票テンプレートのコンパイル 20](#_Toc408305018)

[5． プレビューの表示 21](#_Toc408305019)

[5.1. プレビュー用テストデータソースの作成 22](#_Toc408305020)

[5.2. プレビューの表示とサンプル画像保存 24](#_Toc408305021)

[6． 出力プログラムの作成 25](#_Toc408305022)

[6.1. 帳票出力に必要なデータ 25](#_Toc408305023)

[6.2. 帳票作成処理の呼び出し 28](#_Toc408305024)

[7． グループ・ヘッダーの利用 30](#_Toc408305025)

[7.1. グループの追加 30](#_Toc408305026)

[7.2. グループヘッダーの設定 32](#_Toc408305027)

[7.3. グループフッターの設定 32](#_Toc408305028)

[8． 国際化対応 34](#_Toc408305029)

[8.1. 帳票テンプレートの命名規則（国際化版） 34](#_Toc408305030)

[8.2. ロケール指定のプログラム例 35](#_Toc408305031)

[8.3. ロケールの指定と帳票テンプレートの名前解決仕様 36](#_Toc408305032)

[9． フォントについて 37](#_Toc408305033)

[9.1. 帳票テンプレートのフォントとPDF出力フォント 37](#_Toc408305034)

[9.2. 利用可能な日本語フォントについて 39](#_Toc408305035)

[10． 値の書式編集について 40](#_Toc408305036)

[10.1. 用意されている書式フォーマットを使用する 40](#_Toc408305037)

[10.2. 式で編集 41](#_Toc408305038)

[11． フィールド定義の一括作成 43](#_Toc408305039)

[11.1. Java Beansより作成 43](#_Toc408305040)

# 本書について

本書は、Nablarchの「帳票ライブラリ」を利用して、帳票アプリケーションを開発するための「開発ガイド」である。

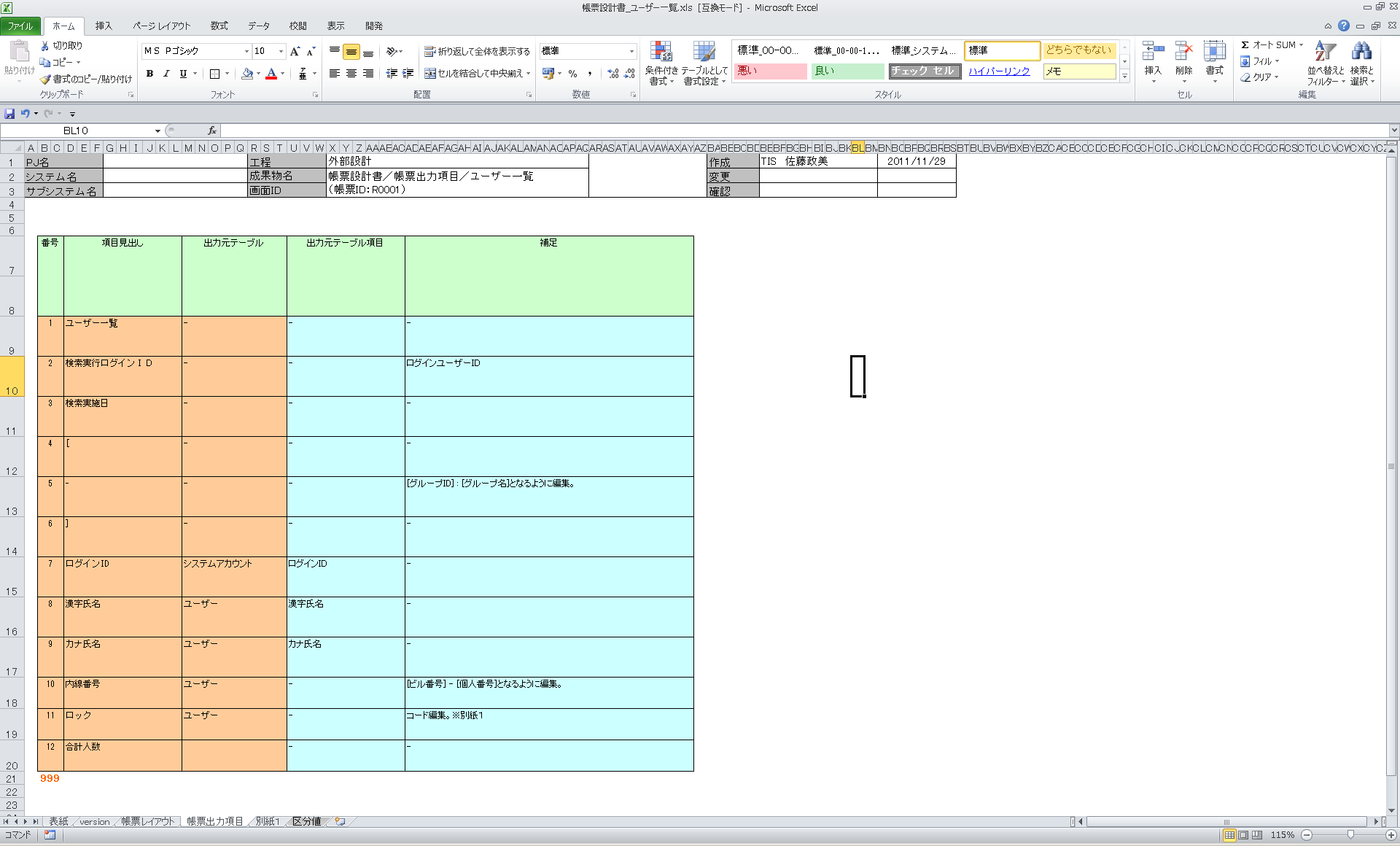
開発フロー、帳票アプリケーションの開発において必要となる帳票テンプレートを作成するためのEcipseプラグインのインストール・操作方法、具体的なプログラミングについて説明する。

# 開発フロー

以下に帳票アプリケーションの開発フローを示す。

1. 設計者が帳票設計書とJaspersoft Studioを用いて帳票テンプレート及びプレビュー画像ファイルを作成する。
2. ①の成果物を顧客・ユーザに説明し合意を得る。
3. 合意を得られた場合、開発者が①の成果物をインプットにし、プログラミング・テスト作業を行う。

③



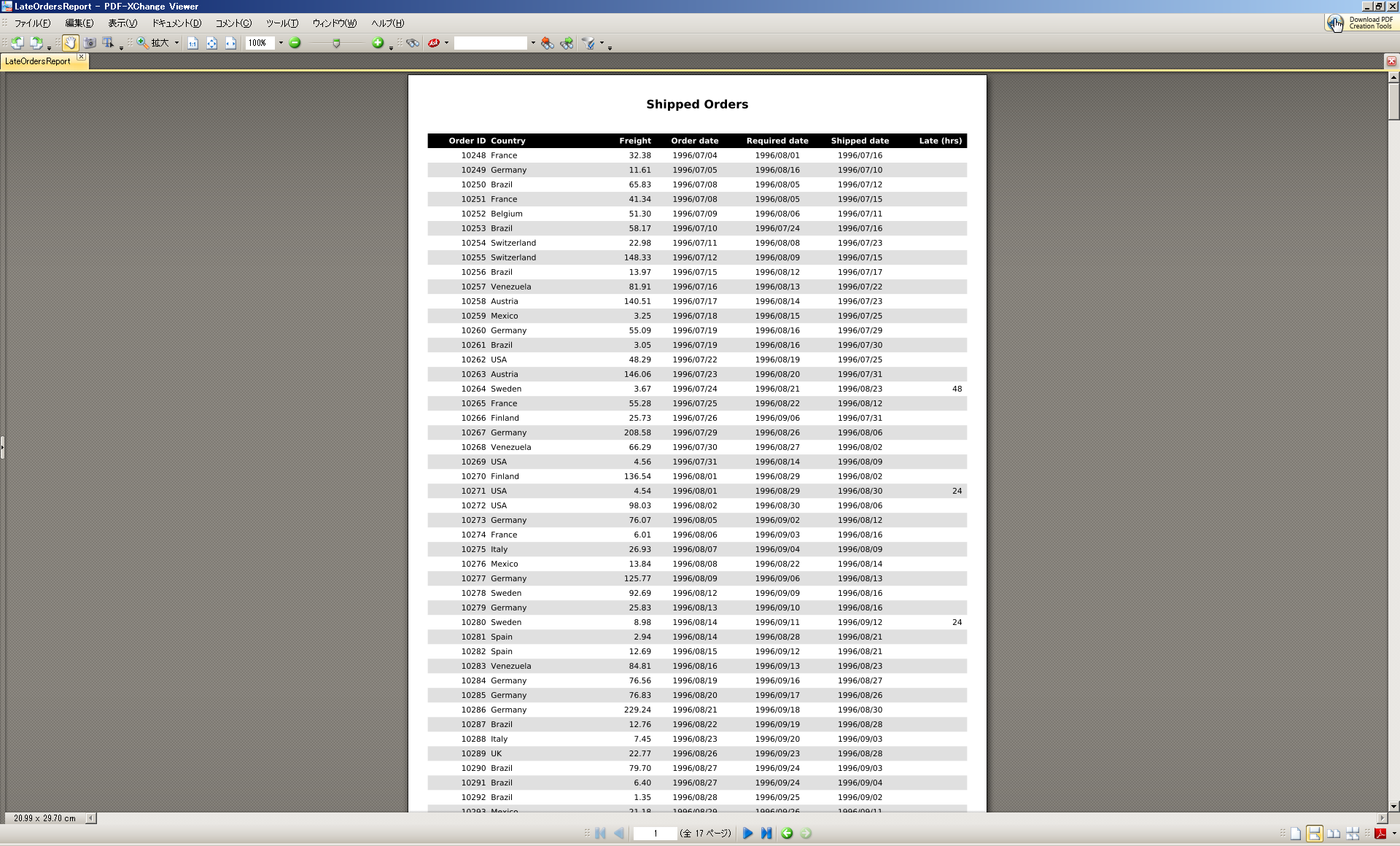
帳票設計書

**ユーザ**

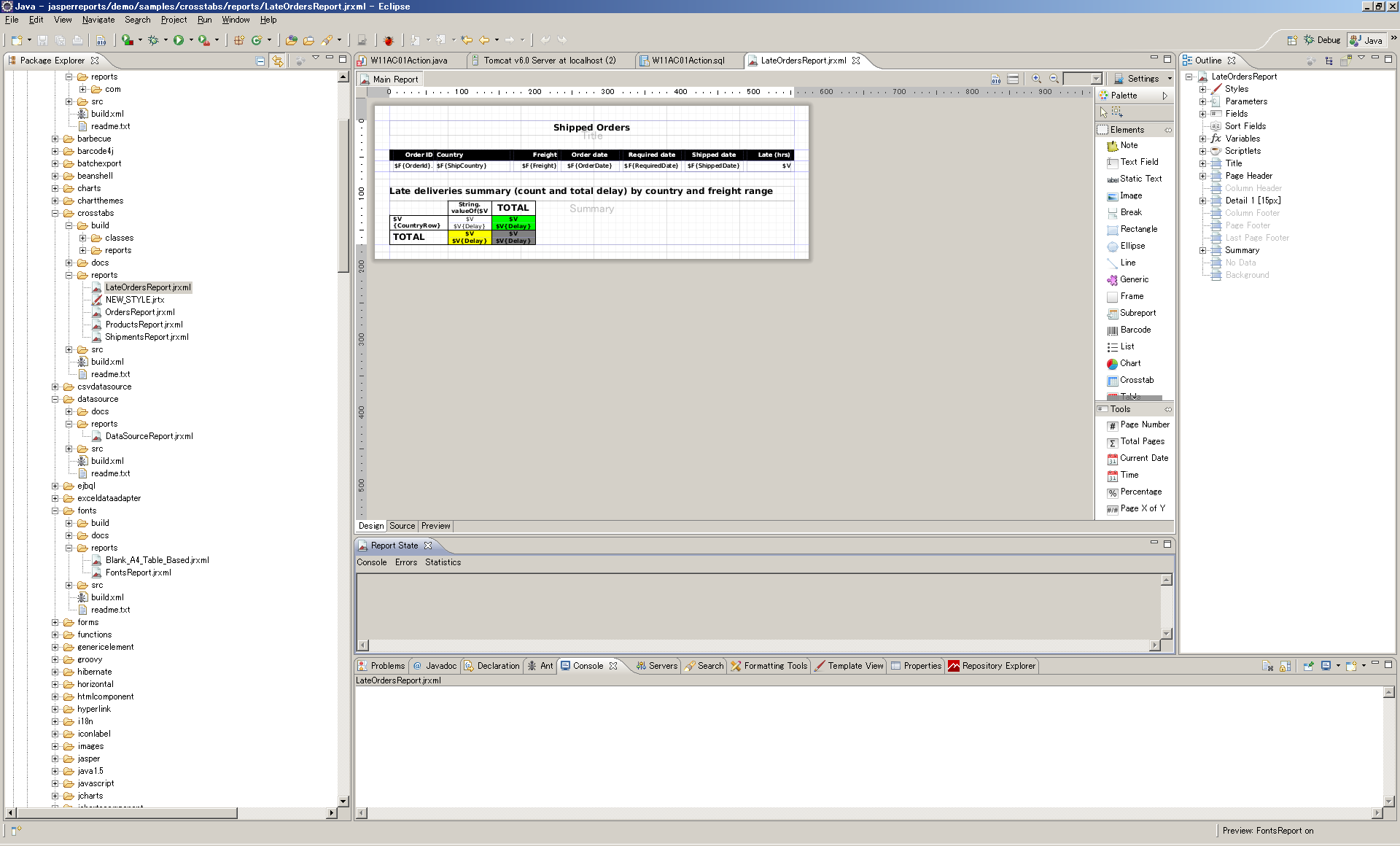
**開発者**



**設計者**



プレビュー画像



帳票テンプレート



## 帳票テンプレートとプログラム

上記の開発フローに登場する帳票テンプレートとは、出力するPDFのデザインを定義するものである。呼び出すプログラム側が帳票テンプレート上に定義された変数に値を渡し、呼び出すことで動的なPDFの生成を可能とする。

帳票テンプレート上でもデータのソートや複雑な書式の編集・式の記述は可能であるが、基本的に複雑な処理・ロジックはプログラムで行い、帳票テンプレート側では単純にその値の表示のみを行うようにすることが望ましい。

# Jaspersoft Studioのインストール

## Jaspersoft Studioのインストール

Eclipseを起動し、メニューの「新しいソフトウエアの追加」をクリックする。

下記の更新サイトを追加し、「Jaspersoft Studio feature」を選択・インストールを行う。

<http://sourceforge.net/projects/jasperstudio/files/updatesite/5.6.2/>

※上記の方法で失敗する場合、下記の方法を実行する。

Jaspersoft Studioプラグインをローカルにダウンロードしてからインストールするという方法となる。

* 1. コマンドプロンプトを起動。下記の２コマンドを実行する。[出力先]には、Jaspersoft Studio プラグインのダウンロードフォルダを指定する。

eclipse.exe -nosplash -verbose -application org.eclipse.equinox.p2.metadata.repository.mirrorApplication –source http://sourceforge.net/projects/jasperstudio/files/updatesite/5.6.1/ -destination [出力先]

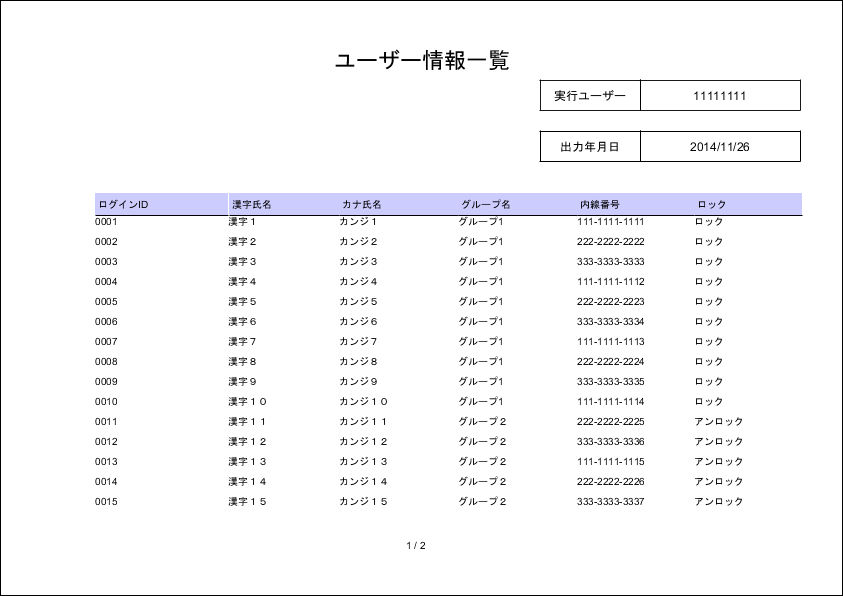
eclipse.exe -nosplash -verbose -application org.eclipse.equinox.p2.artifact.repository.mirrorApplication -source http://sourceforge.net/projects/jasperstudio/files/updatesite/5.6.1/ -destination [出力先]

* 1. Eclipseを起動。メニューの「新しいソフトウエアの追加」をクリックし、表示されたインストールダイアログの「追加」ボタンをクリックする。
  2. リポジトリーの追加ダイアログで「ローカル」ボタンをクリックし、[出力先]のフォルダを指定し、「OK」ボタンをクリック。
  3. 画面の指示に従いインストールを行う。

# 帳票テンプレートの作成

帳票テンプレートの作成を行う。

下図のユーザー情報一覧帳票の開発を例に説明を進める。



## 帳票テンプレートのファイル名と格納場所

標準の帳票テンプレートの命名規則と格納場所は以下の通りである。

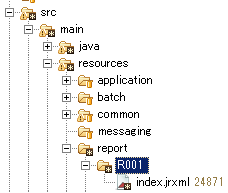
[帳票ベースフォルダ] / [帳票ID] / [帳票テンプレートファイル名]

|  |  |
| --- | --- |
| 項目名 | 説明 |
| 帳票ベースフォルダ | filepath.config キー：file.path.reportで設定された値。 |
| 帳票ID | 帳票毎に割り当てられたID |
| 帳票テンプレートファイル名 | デフォルト名は「index.jrxml」。  国際化対応については[国際化対応](#_国際化対応)を参照のこと。 |

以降の説明は、下記の仕様・設定に従う。

|  |  |
| --- | --- |
| 項目名 | 説明 |
| 帳票ベースフォルダ | filepath.config  file.path.report = classpath:report |
| 帳票ID | R001 |
| 帳票テンプレートファイル名 | index.jrxml |

## 帳票テンプレートの新規作成



帳票テンプレートの格納先フォルダ（右図ではR001フォルダ）を作成し、コンテキストメニューより「新規 – その他」を選択する。

「ウィザードを選択」ダイアログで「Jaspersoft Studio – Jasper Report」を選択し、「次へ」ボタンを押下。

今回はA4横の帳票を作成するために「Report Templates」ダイアログで「Blank A4 Landscape」を選択し、「次へ」ボタンを押下。

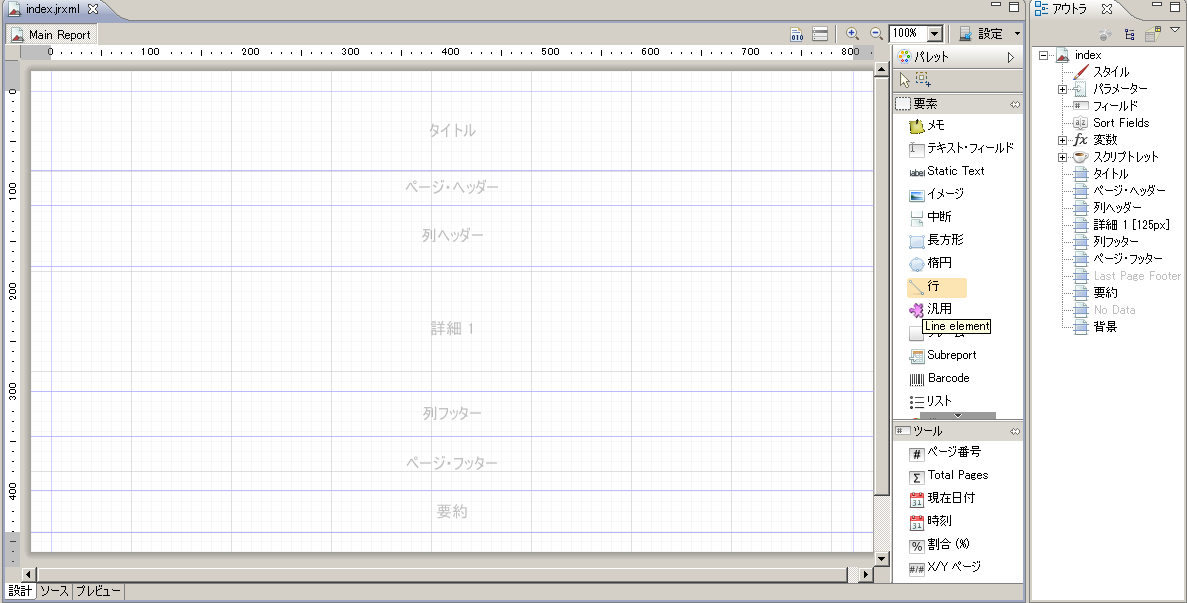
「Report file」ダイアログでファイル名に「index.jrxml」と入力し、「完了」ボタンを押下。

## 各ビューとバンドの概念

### ビューの説明

作成した帳票テンプレートを開くと、下図のような空の帳票及びパレット・アウトラインビューが表示される。

デザイン編集エリア



エディタ上のデザイン編集エリアにパレットビューの要素を配置し、要素ごとにプロパティを設定していく。

### バンドについて

デザイン編集エリアの背景に薄字で表示されている「タイトル」「ページ・ヘッダー」といった区分けされている領域（**バンド**）について説明する。

これらのバンドは帳票を設計する上で出力パターンを分類するものである。

例えば、「タイトル」バンドに置いた要素は、出力する帳票が複数ページにおよぶ場合、1ページ目にだけ表示される。

一方、「ページ・ヘッダー」バンドに置いた要素は、ページ毎に毎回表示される。

「詳細１」と表示されているバンドは、繰り返しデータを表現することが出来る領域である。

「列ヘッダー」には、「詳細１」で出力するデータのヘッダ（列名）を定義する。

## 不要なバンドの削除

今回、初期設定されているバンドの内、

「列フッター」

「要約」

「ページ・ヘッダー」

は使用しないので削除する。

アウトラインビューより該当のバンドを選択し、コンテキストメニューより削除する。

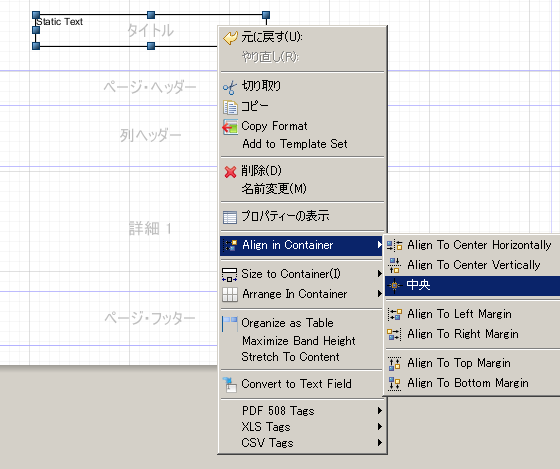
## 要素の配置と操作

### タイトルの設定

「タイトル」バンドに帳票のタイトルを配置する。

今回のタイトルは固定文字であるため、パレットより「Static Text」要素を選択し、「タイトル」バンドに配置する。

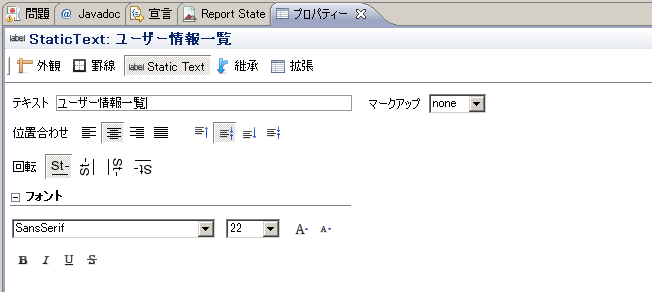
適当なサイズにし、「Static Text」要素を選択した状態でコンテキストメニューより、「Align in Container – 中央」を選択する。



「Static Text」要素のプロパティを修正するため、「Static Text」要素を選択した状態でコンテキストメニューより「プロパティを表示」を選択する。

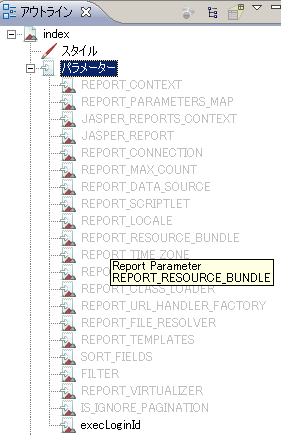
表示されたプロパティビューの「Static Text」タブを選択し、下記のようにプロパティを設定する。

※フォントに関しては注意事項があるため、[フォントについて](#_フォントについて)を参照すること。



|  |  |
| --- | --- |
| 項目名 | 設定値 |
| テキスト | ユーザー情報一覧 |
| 位置合わせ | 水平・垂直ともに中央寄せ |
| フォント | フォントサイズを22 |

### パラメータの設定



帳票テンプレートをプログラム側から呼び出す際に与えることが出来るパラメータを定義する。

このパラメータとは、プログラム側から帳票テンプレートに与える値であり、帳票テンプレート内において

「$P{[変数名]}」で値解決されるものである。

今回はプログラム側で取得した実行ユーザーIDを受け取るパラメータの設定が必要となる。

新しくパラメータを定義するために、アウトラインビューの「パラメータ」を選択し、コンテキストメニューより

「パラメータを作成する」を選択する。

新しく作成されたパラメータ要素を選択し、プロパティを表示する。

プロパティの「名前」に「execLoginId」と入力する。

### 実行ユーザーの見出し要素と値要素の配置



「タイトル」バンドの領域が狭いので適当なサイズに広げる。

「タイトルバンド」の下の青区切り線を下に伸ばす。

「タイトル」バンドに「実行ユーザー」という「Static Text」要素を配置する。

書式を以下のように設定

|  |  |
| --- | --- |
| 項目名 | 設定値 |
| テキスト | 実行ユーザー |
| 位置合わせ | 水平・垂直ともに中央寄せ |
| フォント | フォントサイズを10 |

[パラメータの設定](#_パラメータの設定)で作成したパラメータ「execLoginId」をアウトラインビューよりドラッグ&ドロップで、今配置した「Static Text」要素の近辺に配置する。

ここで配置したパラメータは「テキスト・フィールド」要素となり、自動的に「式」に「**$P{execLoginId}**」という値が入力されている。

「**$P{execLoginId}**」という式により、プログラム側で設定したキー値「execLoginId」とするマップ変数の値解決を行う。

次にこれらの要素の書式設定・整列を行う。

「実行ユーザー」という「Static Text」要素の書式をコピーするために、「実行ユーザー」要素を選択しコンテキストメニューより「Copy Format」を選択する。

「テキスト・フィールド」要素を選択し、コンテキストメニューより「Paste Format」を選択する。

「テキスト・フィールド」の横幅を適当なサイズに広げておく。

要素の整列を行う。

「テキスト・フィールド」要素を選択した状態で、Shiftキーを押しながら見出しの「Static Text」要素を選択する。

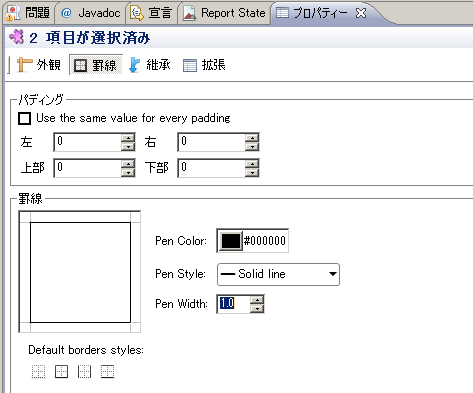
２つの要素が選択された状態で、コンテキストメニューより「Align Component – Join Left」を選択する。

同様に、コンテキストメニューより「Align Component – 上そろえ」を選択する。

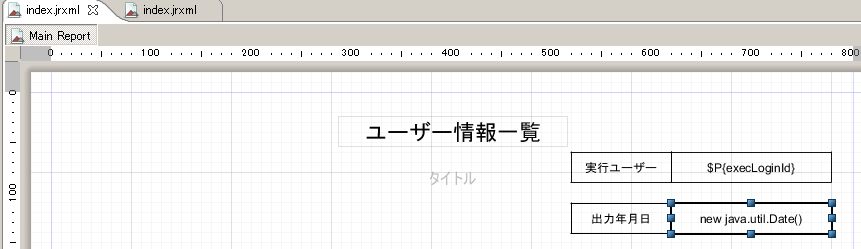
２つの要素に罫線を引く。

２つの要素を選択し、プロパティより「罫線」タブを選択する。

下記の赤丸の部分を押下・入力する。



### 出力年月日の設定



同様に出力年月日を配置する。

日付を出力する「テキスト・フィールド」要素のプロパティは下記のように設定する。

|  |  |
| --- | --- |
| 項目名 | 設定値 |
| 式 | new java.util.Date() |
| パターン | yyyy/MM/dd |

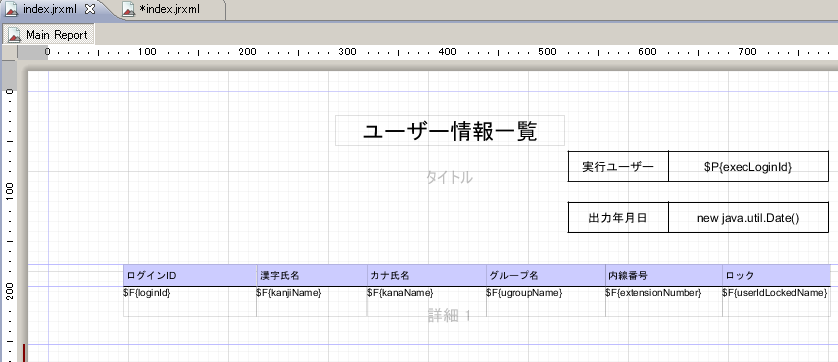
### 繰返しデータの設定

「列ヘッダー」バンドにヘッダー（列名）を「Static Text」要素として配置する。

配置したヘッダー（列名）の「Static Text」要素の高さとバンドの高さを合わせるために、ヘッダー（列名）の「Static Text」要素を全選択し、コンテキストメニューより

「Align In Container – Aligin To Top Margin」を選択。

全選択のまま、次にコンテキストメニューより「Stretch to Content」を選択する。



「詳細」バンドの設定を行う。

「詳細」バンドでは繰り返しデータを表現することになるが、ここでフィールドの設定が必要となる。

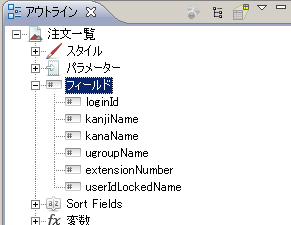
フィールドとは、プログラムより与えられる繰り返しデータに含まれる各レコードの項目である。

この項目名と繰り返しデータの各レコードの項目名がマッピングされる。

繰返しデータとしてＤＢのテーブルデータを用いることが多いため、フィールドの名前はそれらのテーブルのカラム名やEntityのフィールド名と合わせておくことで、プログラム側でのマッピング処理が不要となり実装作業を軽減出来る。

※フィールドの作成はここで説明する方法以外に、Java Beansを読み込んで一括作成する方法がある。[Java Beansより作成](#_Java_Beansより作成)を参照。

アウトラインビューで「フィールド」を選択し、コンテキストメニューより「フィールドの作成」を選択する。



作成されたフィールドを選択しプロパティビューを表示する。

オブジェクトタブの「名前」欄にフィールド名を入力する。

作成するフィールドは以下の６つ。

|  |  |
| --- | --- |
| フィールド名 | 設定値 |
| loginId | ログインID |
| kanjiName | 漢字氏名 |
| kanaName | カナ氏名 |
| ugroupName | グループ名 |
| extensionNumber | 内線番号 |
| userIdLockedName | ロック |

デフォルト設定では、フィールドを「詳細」バンドにドラッグ＆ドロップするとフィールドに対応する見出し要素が自動で作成されるが今回の帳票では不要。

Eclipseのメニューのウィンドウ – 設定 – Jaspersoft Studio – Report Designer の「When a field is drop in the Detail」の値を「Create only the field」に変更する。

作成したフィールドをアウトラインビューで全て選択して、「詳細」バンドにドラッグ＆ドロップする。

ヘッダ（列名）の「Static Text」要素のサイズ及び位置を合わせる。

### ページフッターの設定

最後に「ページ・フッター」バンドに「現ページ数 / 全ページ数」を表示するようにする。

アウトラインビューの「変数」を展開し、「PAGE\_NUMBER」を確認。

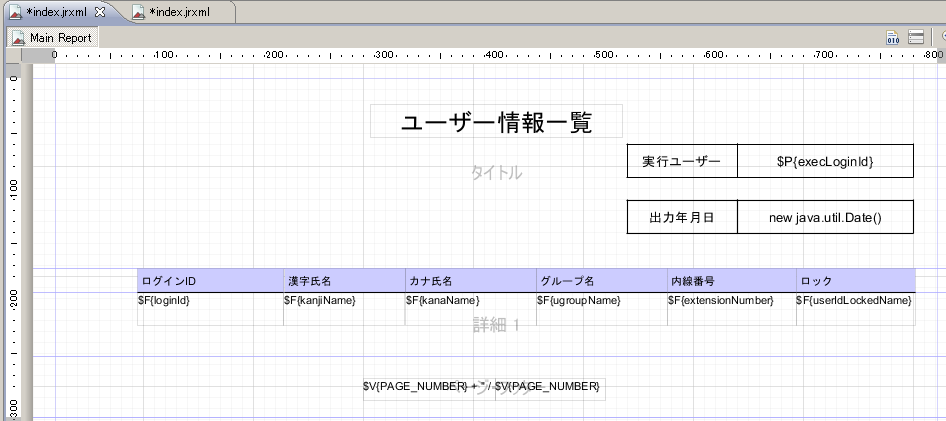
「PAGE\_NUMBER」を「ページ・フッター」バンドに２つドラッグ&ドロップする。

１つ目（現ページ数）の「テキスト・フィールド」要素のプロパティ - 「式」の値を以下のように編集する。

|  |  |
| --- | --- |
| 項目名 | 設定値 |
| 式 | $V{PAGE\_NUMBER} + " / " |
| 位置合わせ | 右寄せ |

２つ目（全ページ数）の「テキスト・フィールド」要素のプロパティ - 「Evaluation Time」を以下のように設定。

|  |  |
| --- | --- |
| 項目名 | 設定値 |
| Evaluation Time | Report |
| 位置合わせ | 左寄せ |



Evaluation Timeを変更することで、式を評価するタイミングを指定することが出来る。

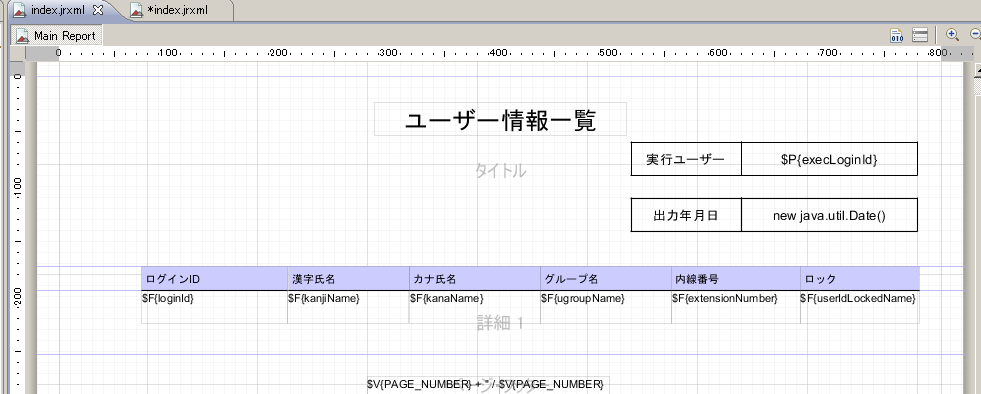
今回の例でいうと、1つ目の「テキスト・フィールド」要素には現ページ数が必要なので「Now」を、２つ目には全ページ数が必要なので、全ページ出力完了時のタイミングを表す「Report」を指定する。

## 帳票テンプレートのコンパイル

作成した帳票テンプレート(jrxml)をコンパイルする。エディタの右上（赤丸）のボタンを押下する。

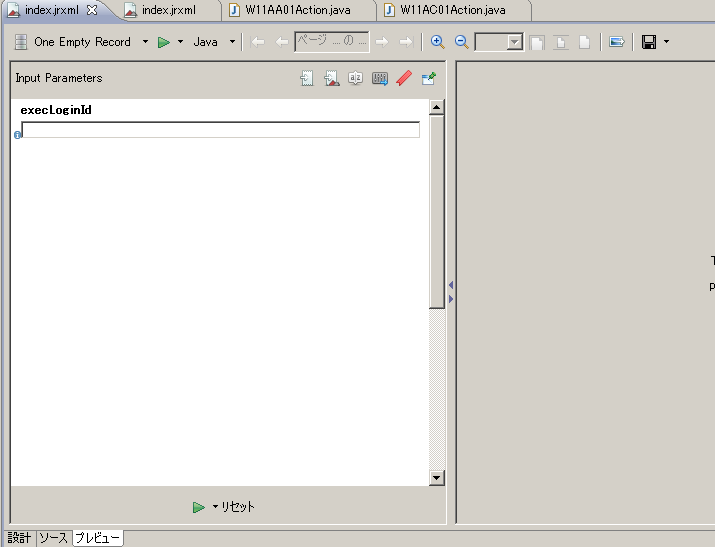
コンパイルに成功すると、index.jasperというファイルがindex.jrxmlと同フォルダに作成される。

実際にプログラム側で指定する帳票テンプレートはこの「\*.jasper」となる。



# プレビューの表示

作成した帳票テンプレートのプレビュー表示を行う。



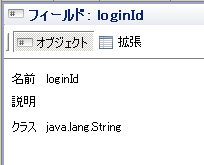
エディタ下部のプレビュータブを選択する。

左ペインに「Input Parameter」が表示され、パラメータ「execLoginId」の入力欄が表示される。

「詳細」バンドがなく繰り返しデータが不要で、パラメータのみの帳票テンプレートの場合は、この「Input Parameter」の入力のみでプレビューを表示することが出来る。

今回は「詳細」バンドに繰り返しデータを表現する必要があるため、プレビュー用の繰り返しデータ（データソース）を用意する。

## プレビュー用テストデータソースの作成



Excelで下記のようなテストデータを作成し、USER\_LIST.xlsxとして保存する。

※「セルの書式設定」で「文字列」を設定すること。

これは今回作成したフィールドのクラスの設定がデフォルトの「java.lang.String」になっているためである。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| loginId | kanjiName | kanaName | ugroupName | extensionNumber | userIdLokedeName |
| 0001 | 漢字１ | カンジ１ | グループ1 | 111-1111-1111 | ロック |
| 0002 | 漢字２ | カンジ２ | グループ1 | 222-2222-2222 | ロック |
| 0003 | 漢字３ | カンジ３ | グループ1 | 333-3333-3333 | ロック |
| 0004 | 漢字４ | カンジ４ | グループ1 | 111-1111-1112 | ロック |
| 0005 | 漢字５ | カンジ５ | グループ1 | 222-2222-2223 | ロック |
| 0006 | 漢字６ | カンジ６ | グループ1 | 333-3333-3334 | ロック |
| 0007 | 漢字７ | カンジ７ | グループ1 | 111-1111-1113 | ロック |
| 0008 | 漢字８ | カンジ８ | グループ1 | 222-2222-2224 | ロック |
| 0009 | 漢字９ | カンジ９ | グループ1 | 333-3333-3335 | ロック |
| 0010 | 漢字１０ | カンジ１０ | グループ1 | 111-1111-1114 | ロック |
| 0011 | 漢字１１ | カンジ１１ | グループ２ | 222-2222-2225 | アンロック |
| 0012 | 漢字１２ | カンジ１２ | グループ２ | 333-3333-3336 | アンロック |
| 0013 | 漢字１３ | カンジ１３ | グループ２ | 111-1111-1115 | アンロック |
| 0014 | 漢字１４ | カンジ１４ | グループ２ | 222-2222-2226 | アンロック |
| 0015 | 漢字１５ | カンジ１５ | グループ２ | 333-3333-3337 | アンロック |
| 0016 | 漢字１６ | カンジ１６ | グループ２ | 111-1111-1116 | アンロック |
| 0017 | 漢字１７ | カンジ１７ | グループ２ | 222-2222-2227 | アンロック |
| 0018 | 漢字１８ | カンジ１８ | グループ２ | 333-3333-3338 | アンロック |
| 0019 | 漢字１９ | カンジ１９ | グループ２ | 111-1111-1117 | アンロック |
| 0020 | 漢字２０ | カンジ２０ | グループ２ | 222-2222-2228 | アンロック |

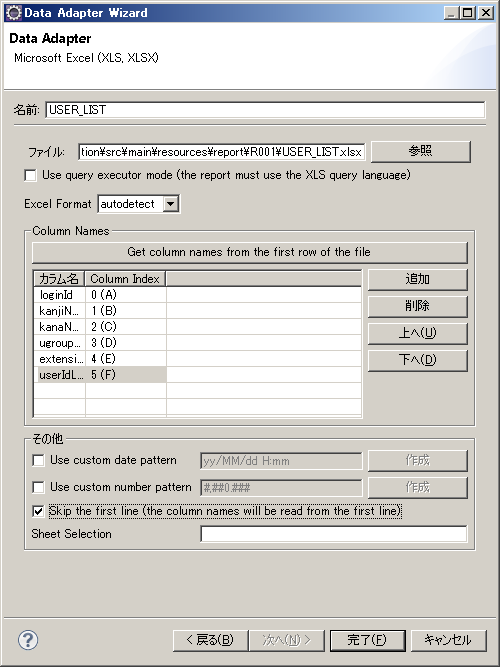
Eclipseのパッケージ・エクスプローラ等で、テストデータを保存するフォルダを選択しコンテキストメニューより

「新規 – その他 – Jaspersoft Studio – Data Adapter」を選択し、次へ。

ファイル名に「USER\_LIST.xml」と入力し、次へ。

次の画面で「Microsoft Excel(XLS, XLSX)」を選択し、次へ。

次の画面で名前に「USER\_LIST」（任意の名前）と入力。



ファイル欄に先ほど作成した「USER\_LIST.xlsx」を指定する。

「USER\_LIST.xlsx」をEclipseワークスペース内に保存し、「参照」ボタンで選択した場合、ファイル入力欄には自動的に相対パスが入力されるが、ここは絶対パスである必要がある。

手動でファイル欄のパスを絶対パスで入力する必要がある。

次に、「Column Names」にある「Get column names from the first row of the file」ボタンを押下する。

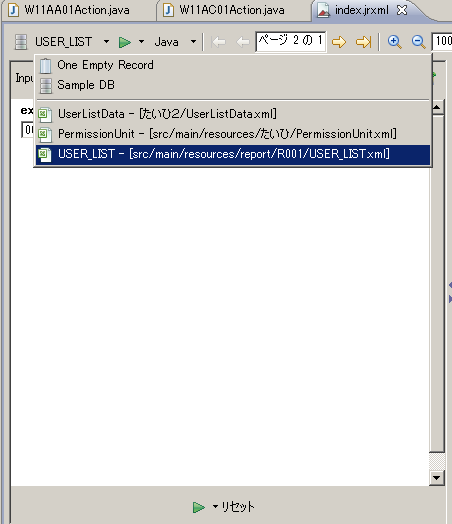
すると、自動的にカラム名の設定が追加される。

列名をスキップするため、「Skip the first line(...　)」のチェックボックスにチェックを入れる。

最後に完了ボタンを押下する。

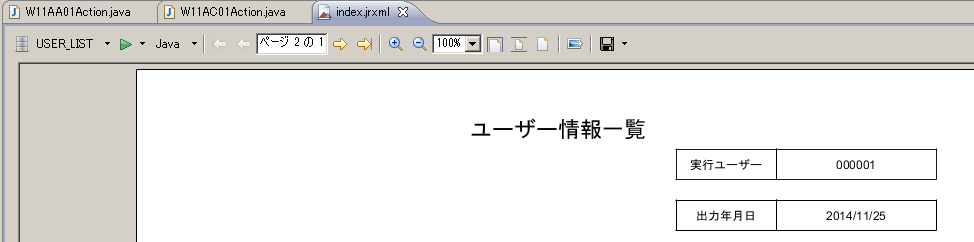
## プレビューの表示とサンプル画像保存

プレビュータブに戻り、「Input Parameter」の「execLoginId」に適当な数値を入力する。



次にデータアダプター選択ボタン（右図、赤丸）により、作成した「USER\_LIST」を選択するとプレビューが表示される。

下図のプレビュー画面の赤丸ボタンを押下することで、プレビュー表示をpng画像として保存することが出来る。



# 出力プログラムの作成

プログラム側では、帳票の作成に必要な業務データなどを取得し、帳票の設定情報とともにそれらのデータを帳票テンプレートにセットし出力メソッドを呼び出す。

## 帳票出力に必要なデータ

帳票テンプレートにセットする出力用データは以下の２つ。

|  |  |
| --- | --- |
| データ種類 | 説明 |
| パラメータ | 帳票テンプレート内で「$P{[変数名]}」で値解決されるデータ。  java.util.Map型又はDTOなどのJava Beans形式を設定可能。 |
| データソース | 「詳細」バンド内の式として「$F{[変数名]}で値解決されるデータ。  java.util.Map型やJava Beansのコレクション、nablarch.core.db.statement.SqlResultSetを設定することが出来る。 |

上記の出力用データに加え、「どの帳票テンプレートを用いるか」、「テンプレートの名前は何か」といった制御情報も必要となる。

出力用データ及び制御情報を「ReportContext」と「ReportParam」に設定し、帳票作成処理の管理を担っているReportManagerに引き渡す。

ReportParamの主な設定

|  |  |
| --- | --- |
| メソッド名 | 説明 |
| setParams | パラメータを設定する。 java.util.Map型又はDTOなどのJava Beans形式のオブジェクトをセットする。 |
| setDataSource | 繰返しデータとなるデータソースを設定する。  データソースオブジェクトを直接設定するのではなく、データソースの形式（java.util.Mapのコレクションや配列、Beanのコレクション、CSV、 SqlResultSetなど）に適したクラスでラップして設定する。  例）  reportParam.setDataSource(new **JRMapCollectionDataSource**(list));  JasperReportsで用意されているラッパークラスに加えて、Nablarchの nablarch.core.db.statement.SqlResultSet型に対応した SqlResultSetDataSourceを用意。 |

　ReportContextの主な設定

|  |  |
| --- | --- |
| メソッド名 | 説明 |
| setReportId | 帳票IDを設定する。  主な用途は帳票テンプレートのファイルパスの解決や出力場所など。 |
| addReportParam | ReportParamオブジェクトを設定する。 |

下記に実装例を示す。

「ReportContext」には帳票IDと

「ReportParam」をセットする。

「ReportParam」には出力用データをセットする。

// 帳票出力処理

// パラメータマップの作成

Map<String, Object> params = **new** HashMap<String, Object>();

params.put("test", “111111”);

ReportParam rp;

if (!searchResult.isEmpty()) {

// データソースとして、searchResultを設定する。

rp = new ReportParam(params, new SqlResultSetDataSource(

searchResult));

} else {

rp = new ReportParam(params);

}

// 帳票コンテキストに帳票IDと帳票パラメータを設定する。

ReportContext rctx = **new** ReportContext(“R001”);

rctx.addReportParam(rp);

## 帳票作成処理の呼び出し

帳票作成に必要な情報を設定された「ReportContext」を「ReportManager」の帳票出力メソッドに渡すことで、帳票ファイルが作成される。

出力形式として、ファイル形式と（java.io.File）とストリーム形式(java.io.InputStream)に対応する。

|  |  |
| --- | --- |
| ReportManager帳票生成メソッド | 説明 |
| File createReport(ReportContext ctx) | ファイル形式で出力。出力場所は「ReportOutputResolver」実装クラスに委譲。 |
| InputStream createReportStream(ReportContext ctx) | ストリーム形式で出力。ストリーム処理の詳細は「ReportExporter」実装クラス#exportStream()メソッド。 |

下記に帳票生成処理の一連のコーディング例を示す。

パラメータはjava.util.HashMap型を、データソースはnablarch.core.db.statement.SqlResultSet型を用いる。

データベースよりレコードを取得後、必要な加工を施したnablarch.core.db.statement.SqlResultSetをデータソースとして使用する。

nablarch.core.db.statement.SqlResultSetの内容 は帳票テンプレートで設定したフィールド名をキー値としたレコード群である。

**loginId kanjiName kanaName　ugroupName extensionNumber userIdLockedName**

**10000 漢字　　　　　カンジ　　　グループ１ 　111-1111-1111 　 ロック**

**SqlResultSet**

**SqlRow**

// 帳票出力処理

Map<String, Object> params = new HashMap<String, Object>();

String loginId = ThreadContext.getUserId();

params.put("execLoginId", loginId);

// 帳票パラメータの作成

ReportParam rp;

if (!searchResult.isEmpty()) {

// データソースとしてsearchResultを設定

rp = new ReportParam(params, new SqlResultSetDataSource(

searchResult));

} else {

rp = new ReportParam(params);

}

// 帳票コンテキストの設定

ReportContext rctx = new ReportContext("R001");

rctx.addReportParam(rp);

File tmpPdf = ReportManager.createReport(rctx);

StreamResponse response = new StreamResponse(tmpPdf, true);

response.setContentType("application/octet-stream");

response.setContentDisposition("ユーザ情報一覧.pdf");

# グループ・ヘッダーの利用

ユーザーのグループ毎に合計人数を表示する修正を加える。

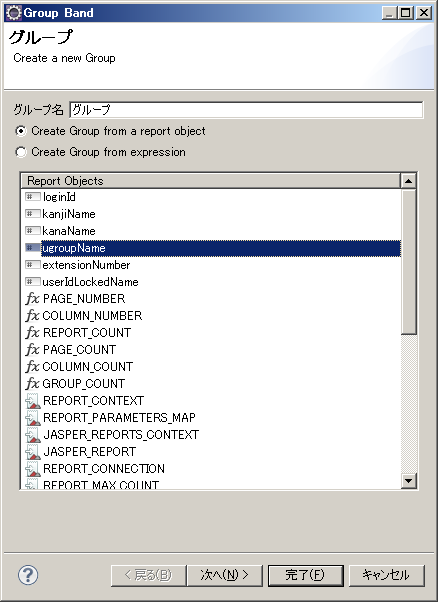
これを実現するために「グループ」バンドを使用する。

## グループの追加

アウトラインビューでルート要素「index」を選択し、コンテキストメニューより「Create Group」を選択する。

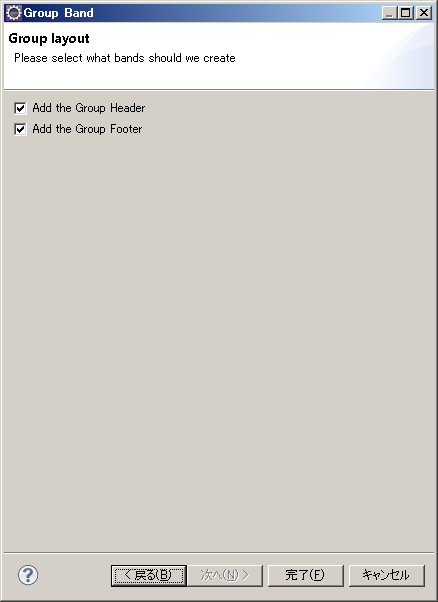
すると、グループバンドダイアログが表示されるので下記の設定を行う。

グループの設定



|  |  |
| --- | --- |
| 項目名 | 設定値 |
| グループ名 | グループ |
| Create Group form a report object | チェック |
| Report Objects | ugroupName |

次へボタン押下。



Group layoutの設定

|  |  |
| --- | --- |
| 項目名 | 設定値 |
| Add the Group Header | チェック |
| Add the Group Footer | チェック |

## グループヘッダーの設定

追加された「グループ・ヘッダー」バンドにグループ名を配置する。

アウトラインビューのフィールドより「ugroupName」を選択し、「グループ・ヘッダー」バンドにドラッグ＆ドロップする。

「TextField Wizard」が表示されるが、「No Calculation Function」が選択されていることを確認して、「完了」ボタンを押下。

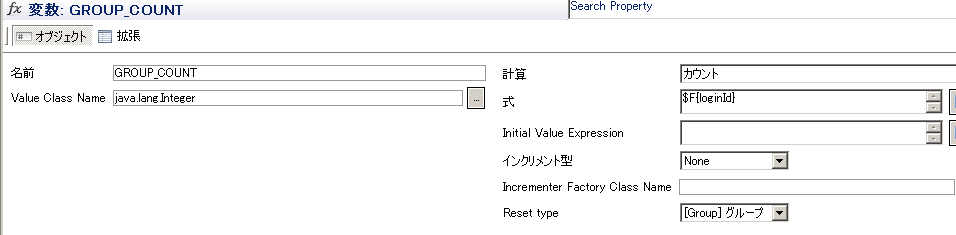
配置した「ugroupName」テキストフィールド要素や「グループ・ヘッダー」バンドの高さを調整する。

## グループフッターの設定

追加された「グループ・フッター」バンドにグループ毎の合計人数を設定する。

アウトラインビューの「変数」を選択し、コンテキストメニューより「Create Variable」を選択。

新しく作成された変数を選択し、下記の設定を行う。



|  |  |
| --- | --- |
| 項目名 | 設定値 |
| 名前 | GROUP\_COUNT |
| Value Class Name | java.lang.Integer |
| 計算 | カウント |
| 式 | $F{loginId} |
| Reset type | [Group]グループ |

アウトラインビューより設定した「GROUP\_COUNT」変数を「グループ・フッター」バンドにドラッグ＆ドロップする。

見出しとして「合計人数」を「Static Text」要素として配置し、各要素のサイズ・位置を調整する。



# 国際化対応

必要な言語分の帳票テンプレートを用意することで実現する。

プログラム側でReportContextのlocaleを指定することで、どの言語の帳票テンプレートを用いるかを決定する。

## 帳票テンプレートの命名規則（国際化版）

|  |  |
| --- | --- |
| ファイル名 | 説明 |
| 帳票テンプレートファイル名 | テンプレートファイル名（デフォルト：index.jrxml）にjava.util.Localeのlanguageとアンダースコアから構成される。  例）  Locale.US →　index\_en.jrxml |

## ロケール指定のプログラム例

// 帳票出力処理

Map<String, Object> params = new HashMap<String, Object>();

String loginId = ThreadContext.getUserId();

params.put("execLoginId", loginId);

// 帳票パラメータの作成

ReportParam rp;

if (!searchResult.isEmpty()) {

// データソースとしてsearchResultを設定

rp = new ReportParam(params, new SqlResultSetDataSource(

searchResult));

} else {

rp = new ReportParam(params);

}

// 帳票コンテキストの設定(ロケールを指定)

ReportContext rctx = new ReportContext("R001", **Locale.US**);

rctx.addReportParam(rp);

File tmpPdf = ReportManager.createReport(rctx);

StreamResponse response = new StreamResponse(tmpPdf, true);

response.setContentType("application/octet-stream");

response.setContentDisposition("ユーザ情報一覧.pdf");

## ロケールの指定と帳票テンプレートの名前解決仕様

プログラム側でロケールを指定しない場合は、「nablarch.core.ThreadContext#getLanguage」の値を使用する。

以下に、帳票テンプレートファイル名「index.jrxml」の名前解決の仕組みを示す。

* 1. ReportContextのlocaleを取得する。未設定の場合は、nablarch.core.ThreadContext#getLanguageを取得する。
  2. 帳票テンプレートファイル名「index.jrxml」と①のlocaleを用いて言語指定の帳票テンプレート名を生成する。（　例：index\_en.jrxml）
  3. ②のコンパイルファイル（　例：index\_en.jasper）が、[帳票ベースフォルダ]/[帳票ID]フォルダ配下に存在する場合は、そのファイルを帳票テンプレートとして用いる。
  4. ③で見つからない場合、デフォルト名（index.jrxml）のコンパイルファイルを帳票テンプレートとして用いる。

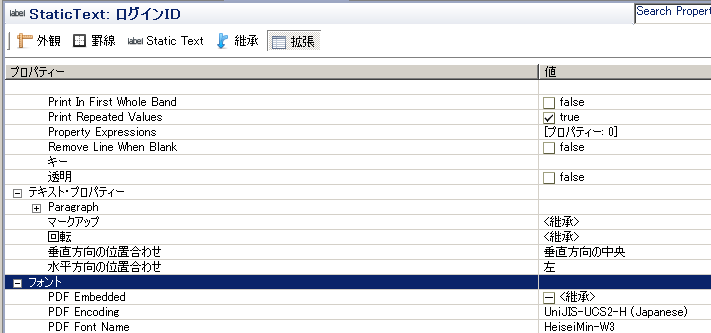
# フォントについて

## 帳票テンプレートのフォントとPDF出力フォント

帳票テンプレート内に配置した「Text Field」「Static Text」要素のプロパティでは「フォント」の種類を設定できる。

しかし、これは帳票テンプレート設計時のみ有効で、PDF出力時には無視される。

PDF出力時のフォントは、各「Text Field」「Static Text」要素の拡張タブにある、「PDF Encoding」「PDF Font Name」で設定を行う。



各「Text Field」「Static Text」要素ごとに設定するのではなく、帳票テンプレート内でデフォルト設定を行いたい場合は、以下の設定を行う。

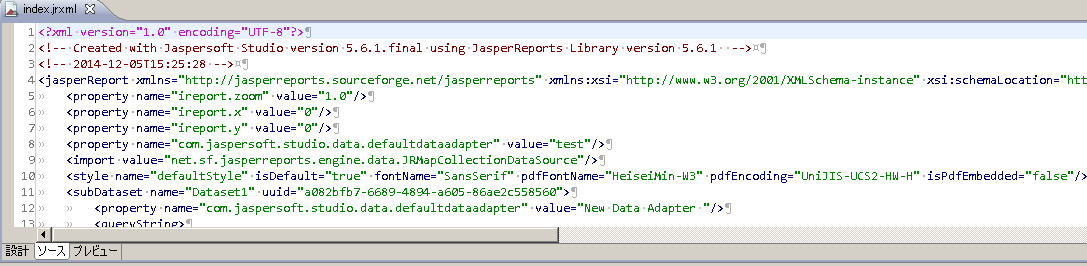
1. アウトラインビューの「スタイル」を右クリック - 「Create Style」をクリックする。この時点で「Style1」という項目が追加される。
2. 帳票テンプレートのソースタブを開き、追加された「Style1」の定義を編集する。

編集前

<style name="Style1"/>

編集後

<style name="defaultStyle" isDefault="true" fontName="SansSerif" pdfFontName="HeiseiMin-W3" pdfEncoding="UniJIS-UCS2-HW-H" isPdfEmbedded="false"/>



## 利用可能な日本語フォントについて

PDF出力で利用可能な日本語フォントとエンコーディングに関しては以下の通り。

利用可能な日本語PDFフォント

|  |  |
| --- | --- |
| フォント名 | 説明 |
| HeiseiMin-W3 | 明朝体 |
| HeiseiKakuGo-W5 | ゴシック体 |

利用可能なPDFエンコーディング

|  |  |
| --- | --- |
| エンコーディング名 | 説明 |
| UniJIS-UCS2-H | Adobe日本語文字のUnicode用エンコーディング（横書き） |
| UniJIS-UCS2-V | UniJIS-UCS2-Hの縦書きエンコーディング |
| UniJIS-UCS2-HW-H | UniJIS-UCS2-Hのうち、プロポーショナル文字のみ半角文字に置き換えるもの |
| UniJIS-UCS2-HW-V | UniJIS-UCS2-HW-Hの縦書きエンコーディング |

# 値の書式編集について

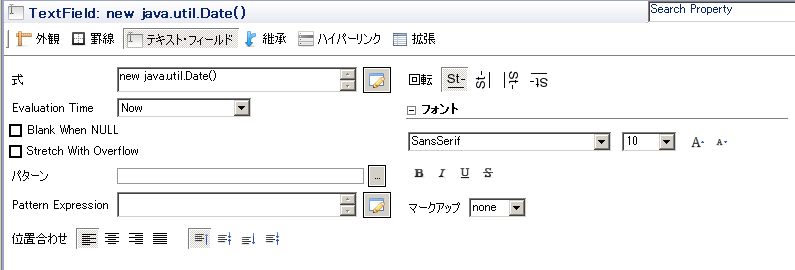
金額や日付、郵便番号といった値の書式フォーマットに関して説明する。

例えば、「1000」を「1,000」、「20150101」を「2015年01月01日」として表示するような場合である。

## 用意されている書式フォーマットを使用する

JasperReportで予め用意されている書式フォーマットを用いる方法がある。

対象の「テキスト・フィールド」要素のプロパティを表示し、「パターン」を選択する。



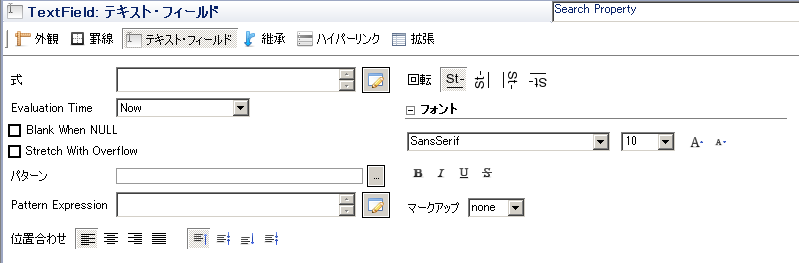
表示されたダイアログで、目的の書式フォーマットを選択する。

テンプレートのフォーマットを参考にして書式フォーマットを調整する。

## 式で編集

予め用意された書式フォーマットでは要件を満たせない場合は、式を編集することで対応する。

対象の「テキスト・フィールド」要素のプロパティを表示し、「式」を選択する。

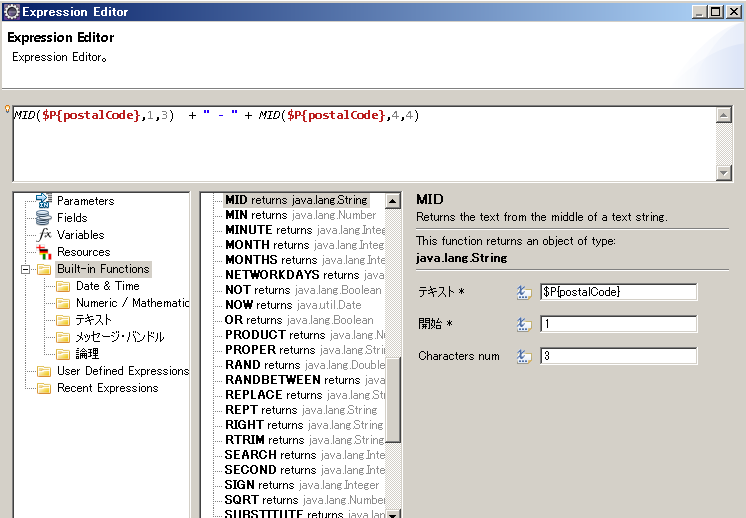


「Parameters」「Fields」よりフォーマット対象の項目を選択・ダブルクリックする。すると、その項目の値を解決する式が貼り付けれらる。

「Built-in Functions」より、値を編集する関数を選択することが出来る。

これらの式を組み合わせることで、書式フォーマットを調整する。

　郵便番号7桁の「3桁 – 4桁」編集の例。

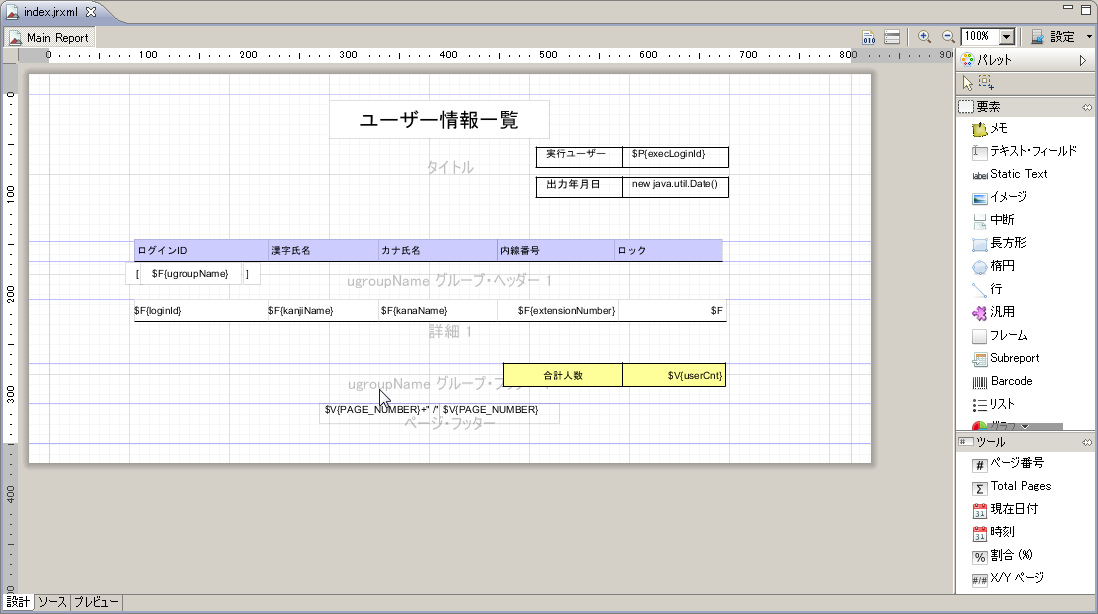


# フィールド定義の一括作成

## Java Beansより作成

フィールドを作成する方法はアウトラインビューで一つずつ作成する以外に、任意のJava Beansを読み込んでフィールドを作成する方法がある。

「DataSet and Query editor dialog」アイコンをクリックする。



「Dataset and Query Dialog」ダイアログで、Java Beanタブをクリックしクラス選択ボタンを押下し対象のJava Beanを選択する。

すると、フィールドの一覧が表示されるので必要なものを選択し、「Add selected field(s)」ボタンをクリックする。

選択したフィールドを確認し、「OK」ボタンをクリックするとフィールドが作成される。

