

## レコード関連

- 前のレコードと値を比較したい。
- 次のレコードと値を比較したい。
- 詳細がない(データがNULLの場合、行を詰めたい。
- 最初のレコードを抽出したい。
- 最後のレコードを抽出したい。
- 1レコードごとに改ページしたい。
- 総レコード数を取得したい。
- レコードの順番はどのように決まる？
- 積算合計やクロス集計などでフィールドを選択できないのはなぜ？
- RecordNumberをグループ毎にリセットしたい。
- 1ページに表示するレコード数を指定したい。
- レコード毎に画像データを表示したい。
- 空白行(レコード)を表示したい。

## グループ関連

- グループの最初のレコードを抽出したい。
- グループの最後のレコードを抽出したい。
- グループヘッダーを毎ページ表示したい。
- グループヘッダーを毎ページ表示したい(グループフッターのみの場合にも表示する)
- グループごとに改ページしたい。
- 動的にグループ化のOn/Offを切り替えたい。
- グループ内件数を取得する。
- 日付でグループ化したときの挙動が怪しい。

## ページ関連

- 総ページ数を取得したい。
- 最初のページ、最後のページを取得したい。
- ページヘッダに詳細の一部を表示したい。
- レポートのページサイズを変更したい。
- ページごとの合計を取得したい。
- 1ページに座標指定(任意の位置に)でオブジェクトを配置する。
- レポートフッターで改ページしたい。
- 指定行で改ページしたい。

## データソース関連

- データソースの置換したい。

## フィールドデータの編集

- 文字列を連結したい。
- 文字列を分割したい。
- 文字列を置換したい。
- 文字列を指定文字数になるまで指定文字で埋めたい。
- 文字列に変換したい。
- 数値を文字列に変換するときに小数点以下の桁数を指定したい。
- 数値を四捨五入したい。

除算した余りを求めたい。

- 変数に値を代入したい。

## フィールド表示形式

- 小数点以下の桁数を指定したい。
- フィールドを非表示にしたい。
- テキストの合成について。
- 文字列を指定位置で改行したい。
- テキストフィールドの改行が思い通りにいかない場合。
- フィールドを罫線で囲みたい。
- 線オブジェクトの表示・非表示を動的に切り替えたい。
- 日付を和暦表示したい。
- テキストオブジェクトの改行の仕方。
- 境界線が消えないようにする。
- フィールドのデータがNullかどうかを判断したい。

## プログラム側からの制御

- 総ページ数を取得する。
- テキストフィールドの値を取得または変更する。
- 式フィールドに値を代入する。
- 式フィールドの式を取得または変更する。
- 動的にグループ条件を変更する。
- 動的にグループをOn/Offしたい。
- レポートに表示しているデータをプログラム側から変更したい。
- サブレポートを取得する。
- 接続先データベース(データソース)を切り替える(その1)。
- 接続先データベース(データソース)を切り替える(その2)。
- 接続先データベース(データソース)を切り替える(その3)。
- CrystalReportViewer でレポートの最後のページまで移動しないで最初のページに合計ページ数を表示する方法。
- CrystalReportViewerのタブの「メインレポート」という文字を変更したい
- CrystalReportViewerのツールバーに表示されているロゴをカスタマイズしたい
- CrystalReportViewerのレポートが表示されている部分のイベントをフックしたい。
- 画像を指定して表示したい。

## 印刷に関して

- プリンタから直接印刷する。
- CrystalReportViewerを使って印刷したい。
- バーコードを出力したい。
- クリスタルレポートの余白をプログラムから設定したい。
- 外字印刷。

## その他

- 製品版とバンドル版の違い。
- VS.NETで実行ファイルにレポートファイル(rptファイル)を埋め込む。
- レポートファイルは実行ファイルに埋め込んだ方が良い？
- Basic構文とCrystal構文の違い。
- CrystalReportsのバージョンアップする際、前の環境を残しておきたい。
- CrystalReportsを用いたアプリケーションの配布について。
- TableModuleとTableDataGateway。
- CrystalReportsの関数のヘルプはないの？
- CrystalReportsで印刷時に文字化けする。

"~"の文字化け。

- 「このフィールドは集計できません」というエラーの意味。
- CrystalReportsをバージョンアップしたら罫線の長さが太くなった。
- データベース構造の変更をレポートに反映させたい。
- 特殊フィールドのレポートタイトルとかレポートコメントって何？
- CrystalReportViewerのツールチップを抑制する。

## 用語

- WhitePrintingRecords
- CurrencyVar
- ドリルダウン
- 条件付き書式設定
- Shared変数
- 続くセクションをアンダーレイ
- データプロバイダ
- レポートファイル
- 積算合計フィールド
- 特殊フィールド
- 式フィールド
- パラメータフィールド
- グループ名フィールド
- SQL 式フィールド
- License Key
- 循環フィールド
- 非循環フィールド
- Chr(13) & Chr(10)

## ADO.NETのよくあるエラーと解決方法

- 追加情報 : Parameter 'PARAM1': No size set for variable length data type: String.
- 追加情報 : ORA-01400: cannot insert NULL into ("PESON\_DB"."PESON"."ADDRESS")
- 追加情報 : ORA-01008: not all variables bound
- 追加情報 : ORA-01036: 変数の名前/数が無効です。
- 追加情報 : No data exists for the row or column.
- 追加情報 : ORA-00920: invalid relational operator
- 追加情報 : ORA-00923: FROM keyword not found where expected
- 追加情報 : ORA-01008: バインドされていない変数があります。
- 追加情報 : ORA-00936: 式がありません。
- 追加情報 : ORA-00918: 列の定義が未確定です。
- 追加情報 : ORA-00933: SQL command not properly ended

## リンク集

- クリスタルレポート関連

## レコード関連

**前のレコードと値を比較したい。**

Previous(フィールド)を使えば、フィールドを指定して一つ前のレコードのデータを参照できます。ただし、最初のレコードなど前のレコードが存在しない場合は評価できないので注意が必要です。

その場合を考慮して、例えばPreviousIsNull(フィールド)と論理積を取る手法が考えられます。

```
//前のレコードと社員情報が一致しない場合にTrueを返すCrystal構文
WhilePrintingRecords;
//<前方>社員情報が存在しない?      and   <後方>社員情報が一致しない?
//   しない/false          and       ×(評価できない)   false
//   する/true             and       する/false         false
//   する/true             and       しない/true        true
(PreviousIsNull({社員情報}) = false) and ({社員情報} <> Previous({社員情報}));
```

[Top](#)

### 次のレコードと値を比較したい,

Next(フィールド)を使えば、フィールドを指定して一つ前のレコードのデータを参照できます。

ただし、最後のレコードなど次のページが存在しない場合は評価できないので注意が必要です。その場合を考慮して、例えばNextIsNull(フィールド)と論理積を取る手法が考えられます。

```
//次のレコードと社員情報が一致しない場合にTrueを返すCrystal構文
WhilePrintingRecords;
//<前方>社員情報が存在しない?      and   <後方>社員情報が一致しない?
//   しない/false          and       ×(評価できない)   false
//   する/true             and       する/false         false
//   する/true             and       しない/true        true
(NextIsNull({社員情報}) = false) and ({社員情報} <> Next({社員情報}));
```

[Top](#)

### 詳細がない(データがNULLの場合、行を詰めたい,

セクションエキスパートで詳細セクションの「空セクションの非表示」をチェックします。

[Top](#)

### 最初のレコードを抽出したい,

式フィールドまたはは条件付き書式設定でOnFirstRecordを使います。

```
//最初のレコードのとき変数cntを0で初期化する。
//そうでない場合、cntをインクリメントする。
numberVar cnt;
if OnFirstRecord then cnt := 0 else cnt := cnt+1;
```

[Top](#)

### 最後のレコードを抽出したい,

式フィールドまたはは条件付き書式設定でOnLastRecordを使います。

[Top](#)

## 1レコードごとに改ページしたい,

---

セクションエキスパートで、詳細セクションの「出力後に改ページ」にチェックします。

[Top](#)

## 総レコード数を取得したい,

---

式フィールドでCount(フィールド)を使います。  
フィールドの出現回数を数えることによってレコードの数を取得します。

パラメータを指定すれば、グループ内件数を取得できるなど色々使い道があります。

```
//WhilePrintingReportsは必要ない  
Count({個人.名前})
```

[Top](#)

## レコードの順番はどのように決まる？

---

レコードの順番は選択SQLのOrder By句によって決定されます。  
Order By句の決定はレポートデザインでグループ化などによるレコードの並べ替えによって行われます。  
したがって、データベースのレコード順ではなく、CrystalReportsのメモリ上にロードされる順番であると言えます。  
ただし、Order By句が何も指定されない場合はデータベースの順番で引き出されることになるでしょう。

ちなみに選択SQLの確認は、メニューから「データベース」 「SQLクエリを表示」で行えます。

[Top](#)

## 積算合計やクロス集計などでフィールドを選択できないのはなぜ？

---

レコード選択式には集計フィールドを含めることはできないため、集計フィールドを利用したフィールドは選択項目に現れません。

(この理由については用語の循環フィールド当たりを参照してください。)

また、レコード選択式では以下の関数を含めることができません。

'PageNumber'、'RecordNumber'、'GroupNumber'、'Previous'、'Next'

[Top](#)

## RecordNumberをグループ毎にリセットしたい,

---

RecordNumberは振り直しができません。  
そのため、代わりとなる式フィールドを用意します。

次のような式フィールドを用意し、RecordNumberの代わりに使います。

```
@レコードカウンターのリセット//非表示にしておく。  
WhilePrintingRecords;  
numberVar RecordCounter:=0;
```

```
//詳細に以下の式フィールドを挿入
@レコードカウンター //非表示に・・・せんでもいいのかな。
WhilePrintingRecords;
numberVar RecordCounter;
RecordCounter := RecordCounter + 1;
```

[Top](#)

### 1ページに表示するレコード数を指定したい。

詳細の改ページ条件に以下の式を挿入します。

ちなみに総レコード数はCount関数で取得できます。

```
WhilePrintingRecords;
numberVar displayCount := 50; // 50件表示
Remainder(RecordNumber, displayCount) = 0 and RecordNumber <> @総レコード数;
```

[Top](#)

### レコード毎に画像データを表示したい。

画像データがすでにデータベースやデータセットなどのデータソースに収まっているなら話は早いです。

画像データの入っているデータベースフィールドをレポートのデザインにドラッグ&ドロップすれば、BlobFieldObjectが作成されます。

この状態で実行すれば、BlobFieldObjectにレコード毎に画像が現れます。

画像データはフォルダなどに収められていてデータベースに入っていない場合はOLEオブジェクトを使用します。

OLEオブジェクトのレポートのデザインにドラッグ&ドロップしたら、そのオブジェクトの書式設定を開き、ピクチャタブの保存場所の項目を選択します。

この項目は条件付書式設定が使えるので、そこに画像のURIを指定します。

もし、データ毎に画像を変更したいのなら、ファイルパスを収めたテーブル(データベースでも、データセットでもよい)を用意し、

そのフィールドを保存場所の条件付書式設定で指定すれば良いでしょう。

[Top](#)

### 空白行(レコード)を表示したい。

無理です。

このような要件は大抵固定行で出力したい場合に出てくるのですが、CrystalReportsは空白行を出力する機能を持ち合わせていません。

そこで、次のように擬似的にデザインすることで空白行を作ります。

1. ページヘッダーで表示したい行数分の枠をデザインします。
2. セクションエキスパートを開き、ページヘッダーの設定で「続くセクションをアンダーレイ」を有効にしておきます。  
(これで次のセクション(詳細)がマージされますので背景として使うことができます。)
3. 詳細でデータ1行分のデザインを行います。

- データの表示テストを行い、データがうまく枠に収まるように詳細の幅を調整します。
- 指定行(レコード)数に達したら改ページするように詳細を設定します。  
(「指定行数で改ページしたい。」を参照。)

[Top](#)

## グループ関連

### グループの最初のレコードを抽出したい。

Previous(フィールド)を用いて、グループで利用しているフィールドの値の変わり目を調べます。

```
WhilePrintingRecords;  
{グループで利用しているフィールド}=Previous({グループで利用しているフィールド})
```

[Top](#)

### グループの最後のレコードを抽出したい。

Next(フィールド)を用いて、グループで利用しているフィールドの値の変わり目を調べます。

```
WhilePrintingRecords;  
{グループで利用しているフィールド}=Next({グループで利用しているフィールド})
```

[Top](#)

### グループヘッダーを毎ページ表示したい。

デザインでグループヘッダーまたはグループフッターを右クリックし、グループの変更オプションから「ページごとにグループヘッダーを出力」にチェックします。  
これにより、グループの詳細がページをまたがるときに、次のページの上にもう一度グループヘッダーを表示することができます。

[Top](#)

### グループヘッダーを毎ページ表示したい(グループフッターのみの場合にも表示する)

通常ページ毎にグループフッターを表示する場合は「ページごとにグループヘッダーを出力」にチェックします。  
これによってグループの詳細が次ページにまたがった場合、再度グループヘッダーが表示されます。しかし、グループフッターのみが次ページにまたがった場合、グループヘッダーは表示されません。これは仕様です。  
これを表示するには以下の手順を取ります。

- 式フィールドでレコードをカウントするフィールドを作り、詳細に置きます。
- グループフッターを2つ作ります。
- グループフッター-aをグループヘッダーと同じ内容にします。
- グループフッター-bを今まで通りのグループフッターとして追加します(同じ内容にします)。
- グループフッター-aの非表示の条件式に、グループフッターが改ページする件数の時に表示するように設定します。

[Top](#)

## グループごとに改ページしたい。

セクションエキスパートで、グループフッターの「出力後に改ページ」にチェックします。

ただし、これだとレポートフッターが改ページされてしまいます(レポートフッターがない場合も空欄ページができると思います)。  
最後のグループフッターでは改ページしないように、上の「出力後に改ページ」の条件付き書式設定で以下のような評価式を指定します。

```
WhilePrintingRecords;  
OnLastRecord <> true;
```

[Top](#)

## 動的にグループ化のOn/Offを切り替えたい。

デザイナー上の設定では不可能です。  
式フィールドによるグループ化を使えば動的にOn/Offを切り替えることが可能となります。

```
WhileReadingRecords;  
// パラメーターフィールド"入力データ"の値が1であるとき、カラムAでグループ化します。それ以外の場合、グループ化しません。  
If {?入力データ} = 1 Then {テーブル.カラムA} Else ""
```

[Top](#)

## グループ内件数を取得する。

式フィールドで自前のカウンターを用意するののも一つの手ですが、これではグループフッター以降でないと実際の件数がわかりません。  
フッターや詳細で直接件数を取得したい場合は式フィールド + Count()関数を用います。

```
//@グループ内件数  
Count({ 1},{ 2});  
// 1 カウント対象のフィールド名、今回は単にレコードの件数を確認したいのでどのフィールドが対象でも構いません。  
// 2 グループ化するフィールド名
```

[Top](#)

## 日付でグループ化したときの挙動が怪しい。

日付型のフィールドでグループ化した場合、デフォルトの設定では1週間単位でグループ化するようになっています。  
グループの変更オプションを開き、共通タブのセクションの出力項目を、毎週 毎日に変更します。

[Top](#)

## ページ関連

### 総ページ数を取得したい。

TotalPageCountで総ページ数を取得できます。

[Top](#)

---

### 最初のページ、最後のページを取得したい。

---

出力状況関数のPageNumberとTotalPageCountを利用すればできます。

例えば以下の式を利用するとレポートの最初と最後のページで真を返し、それ以外では偽となります。

```
WhilePrintingRecords;  
(PageNumber = 1)  
or  
(PageNumber = TotalPageCount);
```

[Top](#)

---

### ページヘッダに詳細の一部を表示したい。

---

メインレポートと同様のデータソースでサブレポートを用意し、サブレポートでその詳細の部分だけを表示するよう(非表示などを利用して)デザインします。そして、そのサブレポートをメインレポートのページフッタに挿入します。

[Top](#)

---

### レポートのページサイズを変更したい。

---

メニュー[ファイル]から[ページ設定]を選び、設定します。

ここで設定される項目はレポートのデザインの一部です。従ってあとで外部から変更することはできません。ただし、印刷する際、プリンタ側で用紙や余白を設定することは可能です。

[Top](#)

---

### ページごとの合計を取得したい。

---

積算合計フィールドを使う手もありますが、変数を使った方が応用が利きます。

サンプルコード参照してください。

```
//ページヘッダーに以下の式フィールドを挿入  
@合計のリセット//非表示にしておく。  
WhilePrintingRecords;  
numberVar TotalValue:=0;  
  
//詳細に以下の式フィールドを挿入(合計をカウントする場所なので詳細とは限らない)  
@合計のカウント//非表示にしておく。  
WhilePrintingRecords;  
numberVar TotalValue;  
TotalValue := TotalValue +{値}  
  
//ページフッターに以下の式フィールドを挿入  
@合計の出力  
WhilePrintingRecords;
```

```
numberVar TotalValue;
```

[Top](#)

## 1ページに座標指定(任意の位置に)でオブジェクトを配置する。

オブジェクトを詳細やグループヘッダー、フッターなどのセクションに関係なく、ページ全体から見た座標指定で配置したいことがあります。

例えば会社のロゴや背景画像を任意の場所に表示したいことがあります。詳細や、グループヘッダー、フッターはレコード数により位置がずれるので配置する場所として妥当ではありません。通常ロゴなどは、位置が固定されるレポートヘッダー、ページヘッダー及びそれらのフッターといったような場所に挿入します。

しかし、その方法では理想のデザインにならず、ページ内に座標を指定して任意の場所にオブジェクトを配置したいことが考えられます。

しかし、クリスタルレポートはページ内で座標指定はできません・・・。

そこで以下のようなデザインにすることで擬似的に位置決めを行います。

1. ページヘッダーをページの大きさの限界まで広げ、1ページ分の余白を作ります。  
ページの基本構成はページヘッダーとページフッターなので、広げられる限界はページフッター分を引いた大きさです。ページヘッダーをページ全体に広げる場合はページフッターを削除(非表示に)します。
2. ページヘッダーの任意の場所のオブジェクトを配置します。
3. このままでは、レポートにページヘッダーしか表示されないのので、セクションエキスパートを開き、ページヘッダーの設定で「続くセクションをアンダーレイ」にチェックします。

すると、他のセクションとマージされ、オブジェクトを座標指定しているように見せかけることができます。

ページヘッダー分の領域がなくなることになるので、他のセクションの余白を調整する必要があります。

[Top](#)

## レポートフッターで改ページしたい。

レポートフッターは改ページできないため、詳細を分割して代用します。  
例えば詳細aと詳細bを用意し、詳細bをレポートフッター用にします。  
最後のページきたら詳細bを表示し、それ以外では表示しないようにします。

セクションエキスパートで詳細bの非表示設定の条件付書式設定で以下の式を挿入します。

```
WhilePrintingRecords;  
PageNumber <> TotalPageCount;
```

[Top](#)

## 指定行で改ページしたい。

次の二つの式フィールドを作成し、@行数の初期化をページヘッダーに、@行数のカウントを詳細に

挿入します。  
また、それぞれフィールドのプロパティで非表示にしておきます。

つづいてセクションエキスパートを開いて、改ページ条件を詳細の「出力後に改ページ」の条件式に記述します。

```
// @行数の初期化      ページヘッダーに挿入する
WhilePrintingRecords;
numberVar line := 0;

// @行数のカウント   詳細に挿入する
WhilePrintingRecords;
numberVar line;
line := line + 1;

// 改ページ条件     詳細の「出力後に改ページ」の条件式に記述する
WhilePrintingRecords;
numberVar line;
line = X // Xは表示したい行数
```

[Top](#)

## データソース関連

### データソースの置換したい。

フィールドエクスプローラのデータベースの欄を右クリック 「データソースの保存場所の設定」から行います。  
置換先がないカラムは型が同じであるならばマージ先を指定できます。

[Top](#)

## フィールドデータの編集

### 文字列を連結したい。

「+」もしくは「&」で結びます。  
ただし、CrystalReportsのバージョンが古い場合、「&」が使えない可能性があります。

```
WhilePrintingRecords;
"文字列の" + "連結";
// 文字列の連結
```

[Top](#)

### 文字列を分割したい。

左から何文字分を取得したい場合はLeft(文字列, 文字数)  
右から何文字分を取得したい場合はRight(文字列, 文字数)  
指定位置から何文字分取得したい場合はMid(文字列, 開始位置, 文字数)

を使います。

[Top](#)

---

### 文字列を置換したい。

---

Replace()を利用します。

```
Replace("abcabcabc", "b", "a");
// aacaacaac

// 変換を複数回行いたい場合は一度変数に格納する方が無難である。
stringVar str;
str := Replace("abcabcabc", "b", "a");
// aacaacaac
Replace(str, "c", " a");
// aaaaaaaaa
```

[Top](#)

---

### 文字列を指定文字数になるまで指定文字で埋めたい。

---

Right()またはLeft()を応用して使います。

本来Right()は指定された文字列の右端から、指定された文字数分の文字を取り出す関数です。これを利用して取り出す前に埋め込む文字を左側に結合しておきます。

```
//10桁になるまで0を埋め込みたい

//左側に埋め込む
stringVar str;
str := "123";
Right("0000000000" & str, 10);
//00000000123

//右側に埋め込む
Left(str & "0000000000", 10);
//1230000000
```

[Top](#)

---

### 文字列に変換したい。

---

ToText()を使います。

論理値フィールドや数値フィールド、日付フィールドの値はToText()によって文字列に変換できます。

```
WhilePrintingRecords;
stringVar str;
numberVar num;

str := "文字列";
num := 123;

// "文字列" と 数値123を文字列に変換して連結する
str := str + ToText(num);
```

```
// "文字列123"
```

[Top](#)

---

### 数値を文字列に変換するとき小数点以下の桁数を指定したい。

---

CStr()を使います。

CStr()は機能的にはToText()といっしょで様々な型を文字列に変換する関数ですが、オプションで様々な変換形式を指定できます。

```
WhilePrintingRecords;  
stringVar str;  
numberVar num;  
  
num := 123;  
// 小数点以下を表示しない  
str := CStr(num, 0);
```

[Top](#)

---

### 数値を四捨五入したい。

---

Round(x, y)を使います。

```
Round(1.23456)  
//1 を返す  
  
//2番目の引数を指定した場合、指定した桁数で丸め込まれる。  
Round(1.23456, 2)  
//1.23 を返す。
```

[Top](#)

---

### 除算した余りを求めたい。

---

Remainder(分子, 分母)を使います。

```
Remainder(12,5)  
//2 を返す。  
  
Remainder(16,5)  
//1 を返す。
```

[Top](#)

---

## 変数に値を代入したい,

---

式フィールドなどで変数に値を代入する場合、「:=」演算子を使います。

```
numberVar num := 0;
```

[Top](#)

## フィールド表示形式

### 小数点以下の桁数を指定したい,

---

「フィールドの書式設定」 「数値」タブ 「ユーザ設定」 「数値」タブ 「小数点以下の桁数」で指定します。

もし、動的に桁数を変えたいならば条件付き書式設定で以下のサンプルのようなコードを記述します。

```
WhilePrintingRecords;
//指定したフィールドの値が0なら小数点以下桁数を表示しない、0以外なら第2位まで表示
if {フィールド名} = 0 then 0 else 2;

////////////////////////////////////
WhilePrintingRecords;
//指定したフィールドの値が小数点以下の値を含まないなら小数点以下を表示しない、含むなら第2位まで表示
if ({フィールド名} - RoundUp({フィールド名}, 0)) = 0 then 0 else 2;
```

[Top](#)

### フィールドを非表示にしたい,

---

設定したいフィールドを選択 「フィールドの書式設定」 「共通」タブ 「非表示」チェックボックスをチェックします。

[Top](#)

### テキストの合成について,

---

テキストフィールド内にフィールドをドラッグ&ドロップするとテキストが合成されます。

- 例.
- ・"[ ]" このフィールドに、"km/h"という値を持つフィールド(例えばフィールド名:Unit)をドラッグ&ドロップ
  - ・デザイナー上、"[Unit]"のように表示されます
  - ・レポートを実行すると、"[km/h]"として表示されます

[Top](#)

## 文字列を指定位置で改行したい。

---

テキストフィールドに挿入する文字列が長い場合、複数行に分けたい場合があります。しかし、OSにより改行の仕方が違うので思い通りに改行できないことがあります。以下のコードは指定した文字数で改行を行うものです。ただし、プロポーショナルフォントや半角/全角の違いの影響を受けます。

```
@10文字単位で改行するフィールド
WhilePrintingRecords;
stringVar str;
stringVar res;
numberVar i;

str := {文字列};
// 対象の文字列から1文字ずつ抽出する。
for i := 1 to Length(str) step 1 do(
res := res + str[i];
// 10文字単位で改行コード(chr(13) + chr(10))を埋め込む。
if Remainder(i,10) = 0 then res := res + chr(13) + chr(10) else res;
);

stringVar res;
```

[Top](#)

## テキストフィールドの改行が思い通りにいかない場合。

---

通常、"複数行に出力"にチェックを入れていれば、フィールドの幅に収まりきらなかった場合に自動で改行されます。しかし、チェックしたにも関わらず変なところで改行される場合があります。これはUSP10.DLLのバージョンの違いにより発生します。USP10.DLLのバージョンはOSによって違います(以下のページを参照)。

[http://www.agtech.co.jp/support/faq/crystal\\_reports/CR\\_all/cr\\_general/20031106005.html](http://www.agtech.co.jp/support/faq/crystal_reports/CR_all/cr_general/20031106005.html)

上記ページによるとWindows2000のUSP10.DLLが最も正常な改行を行っているようで、自分のマシンWin2000以外の場合、Win2000のUSP10.DLLに入れ替えれば直るらしいです。ただし、USP10.DLLは一つのマシンに一つとは限らないので、アプリケーションに影響を与えているUSP10.DLLはどれに当たるのか調べなければなりません。

[Top](#)

## フィールドを罫線で囲みたい。

---

通常、罫線には線オブジェクトを使いますが、表示する内容によって幅が変動する場合ずれてしまいます。そういうときは、フィールドの書式設定の境界線タブを開き、線のスタイルで指定します。

条件付き書式設定を行えるので結構汎用できる項目です。

[Top](#)

## 線オブジェクトの表示・非表示を動的に切り替えたい。

---

線オブジェクトやボックスオブジェクトには条件付き書式設定がないので、動的な設定変更は無理です。

そこで、セクションを分けて、そこに線オブジェクトだけを配置し、セクションの条件付き書式設定で表示・非表示を切り替えることにより対処します。

もし同じ行に複数の線オブジェクトを配置したい場合は、同様にセクションを分け、前のセクションで「続くセクションをアンダーレイ」にチェックすれば可能です。

[Top](#)

---

## 日付を和暦表示したい。

---

フィールドの書式設定を開き、[日付]タブを選択する。スタイルのユーザー設定ボタンを押すとカスタムスタイルを指定できるので、そこでカレンダーの種類を和暦に変えます。年号（平成など）をつけたい場合は前置記号に書きます。

[Top](#)

---

## テキストオブジェクトの改行の仕方。

---

通常、CrystalReportsのフィールドオブジェクトの値はプログラム側(.NET Framework)から直接そのオブジェクトを指定することにより変更することができます。

しかし、CrystalReports9のDeveloper又はAdvanced Editionsではこの方法では改行コードを受け付けません。式フィールドを使う必要があります。

<http://support.businessobjects.com/library/kbase/articles/c2015089.asp>

[Top](#)

---

## 境界線が消えないようにする。

---

境界線はフィールドのデータが空白やNullの場合に消えてしまいます。消えないようにするにはそのフィールドを以下のような式フィールドに置き換えます。

```
WhilePrintingRecords;  
If Isnull({フィールド}) = true then  
// 復帰文字(\r)に置き換える  
chr(13)  
else  
{フィールド}
```

[Top](#)

---

## フィールドのデータがNullかどうかを判断したい。

---

IsNull({フィールド名})で判断できます。

[Top](#)

---

## プログラム側からの制御

### 総ページ数を取得する。

このサンプルコードを実行する前にログオン情報やパラメータフィールドがあるならちゃんと値を設定しておかなければなりません。

(参考:[http://support.businessobjects.com/library/kbase/articles/c2013559.asp?ref=devzone\\_net\\_archive](http://support.businessobjects.com/library/kbase/articles/c2013559.asp?ref=devzone_net_archive))

```
//reportはレポートのインスタンス
//VB.NET
Dim intPages As Integer

'CrystalReportが返す全ページ数を得る
intPages = report.FormatEngine.GetLastPageNumber(New CrystalDecisions.Shared.ReportPageRequestContext)

//C#.NET
CrystalDecisions.Shared.ReportPageRequestContext req =
new CrystalDecisions.Shared.ReportPageRequestContext();

int intPages = report.FormatEngine.GetLastPageNumber(req);
```

[Top](#)

### テキストフィールドの値を取得または変更する。

フィールドオブジェクト名はレポートのデザインでフィールドを右クリックして「テキストの書式」を開き、「共通」タブのオブジェクト名を参照します。

```
CrystalDecisions.CrystalReports.Engine.TextObject txtObj;
txtObj=rptObj.ReportDefinition.ReportObjects["フィールドオブジェクト名"] as CrystalDecisions.
txtObj.Text = "なんちゃらかんちゃら";
```

[Top](#)

### 式フィールドに値を代入する。

サンプルを参照してください。

#### 注意点

その1、式フィールド名とは新規に式を作成する際与える名前のことです。  
その2、式フィールドのTextプロパティの内容はあくまで式であるのでCrystalReportsの式構文チェックをパスするような内容にしなければなりません。

```
//reportはレポート名
report.DataDefinition.FormulaFields.Item("式フィールド名").Text = "'なんちゃらかんちゃら'"
//'で囲っていることに注意
//"なんちゃらかんちゃら"ではエラーとなる。
//数値の場合はそのまま大丈夫。
```

## 式フィールドの式を取得または変更する。

サンプルを参照してください。

```
CrystalDecisions.CrystalReports.Engine.FormulaFieldDefinition field;
field = rptObj.ReportDefinition.ReportObjects["フィールドオブジェクト名"] as CrystalDecisions
field.Text = "なんちゃらかんちゃら";

// 条件に合致するフィールドを検索したい場合、以下のようにしてもOKよ。

foreach(CrystalDecisions.CrystalReports.Engine.FormulaFieldDefinition field in rptObj.DataDefi
{
    if(field.ValueType == CrystalDecisions.Shared.FieldValueType.StringField)
    {
        System.Console.WriteLine(field.Text);
    }
}
```

## 動的にグループ条件を変更する。

式フィールドを利用することで動的にグループ条件を変更できます。

サンプルを説明します。

まずフラグになる式フィールド (@フラグ) を用意します。

次にグループ条件となる式フィールド (@動的グループ) を用意し、このフィールドでグループを作ります。

そして、プログラム側でフラグを制御することでグループ条件を切り替えます。

```
//@フラグ
1

//@動的グループ
WhileReadingRecords; // WhilePrintingRecordsでないことに注意
if {@フラグ} = 1 then
{名前}
else
{住所}
```

## 動的にグループをOn/Offしたい。

データベースのフィールドを直接指定するやり方ではできません。

グループ条件を変更するやり方と同じように式フィールドを利用します。

まずフラグになる式フィールド (@フラグ) を用意します。

次にグループ条件となる式フィールド（@動的グループ）を用意し、このフィールドでグループを作ります。

そして、プログラム側でフラグを制御することでOn/Offを切り替えます。  
else以下は文字列型の時しか使えませんので必要に応じて消してしまってもかまわないと思います。

```
//@フラグ
1

//@日付によるグループ
WhileReadingRecords; // WhilePrintingRecordsでないことに注意
if {@フラグ} = 1 then
{名前}
else
''
```

[Top](#)

### レポートに表示しているデータをプログラム側から変更したい。

例えば、タイトルなどは表示する内容によって変えたい場合があります。  
そういったときはパラメータフィールドを使います（テキストフィールドや式フィールドでも行えます）。

静的なパラメータフィールドを新規作成します。  
オプションはデフォルトのままでもかまいません。

ただし、レポートを表示するときデフォルト値がないまたはプログラム側からパラメータ値が指定されていない場合、  
パラメータの入力を求めるダイアログが表示されます。

以下のようなコードを利用して必ずパラメータ値を指定しておくか、予めデフォルト値を設定しておくべきです。

```
// rptObjはレポートオブジェクト
rptObj.SetParameterValue("レポートタイトル", "クリレポチップス集");
```

[Top](#)

### サブレポートを取得する。

サンプルを参照してください。

```
// rptはレポートドキュメントオブジェクト
SubreportObject subRptObj;
subRptObj = rpt.ReportDefinition.ReportObjects["SubReport1"] as SubreportObject;
ReportDocument subRpt = subRptObj.OpenSubreport(subRptObj.SubreportName);

//又は以下のようにする。こっちの方がスマート
SubreportObject subRptObj;
subRptObj = rp.ReportDefinition.ReportObjects("SubReport1")
```

---

### 接続先データベース(データソース)を切り替える(その1).

---

データベースに接続する方法は3種類あります。

```
CrystalDecisions.CrystalReports.Engine.ReportDocument report;
//絶対パスが無難
string path = Application.StartupPath + @" \ report.rpt";
report.Load(path);

//"ユーザ名,パスワード,サーバー名,データベース名の順
report.SetDatabaseLogon( "oraUser", "oraPassword", "oraSever", "" );
report.Database.Tables[0].Location = "oraUser" + "." + report.Database.Tables[0].Location;
```

---

### 接続先データベース(データソース)を切り替える(その2).

---

```
CrystalDecisions.CrystalReports.Engine.ReportDocument report;
//絶対パスが無難
string path = Application.StartupPath + @" \ report.rpt";
report.Load(path);

TableLogOnInfo logOnInfo = new TableLogOnInfo();
logOnInfo = report.Database.Tables[ "Table1" ].LogOnInfo;

ConnectionInfo connectionInfo = logOnInfo.ConnectionInfo;

connectionInfo.DatabaseName = "";
connectionInfo.ServerName = oraServer
connectionInfo.UserID = "oraUser";
connectionInfo.Password = "oraPassword";

report.Database.Tables[ "Table1" ].ApplyLogOnInfo( logOnInfo );
```

---

### 接続先データベース(データソース)を切り替える(その3).

---

```
CrystalDecisions.CrystalReports.Engine.ReportDocument report;
//絶対パスが無難
string path = Application.StartupPath + @" \ report.rpt";
report.Load(path);

report.DataSourceConnections[0].SetConnection( "oraServer", "", "oraUser", "oraPassword" );
```

## CrystalReportViewer でレポートの最後のページまで移動しないで最初のページに合計ページ数を表示する方法

CrystalReportViewerのステータスバーやツールバーには合計ページ数が表示されますが、起動時には「1+」と表示されています。

起動時にちゃんと合計ページ数を表示させたいければ、最初のページまたはレポートフッターで、TotalPageCountを取得してやれば良いです。

例えば以下の式フィールドを最初のページに挿入します。

```
//表示する必要がないので非表示にしておく
WhilePrintingRecords;
ToText(TotalPageCount,0);
```

[Top](#)

## CrystalReportViewerのタブの「メインレポート」という文字を変更したい

タブ自体を消すことはできませんが、タブ名は変更できます。

ビューアにレポートをセットしたあと、以下のサンプルコードで変更します。

```
foreach (Control obj in crystalReportViewer1.Controls)
{
    if (obj.GetType().Name == "PageView")
    {
        foreach (Control obj2 in obj.Controls)
        {
            if (obj2.GetType().Name == "TabControl")
            {
                ((TabControl)obj2).TabPage[0].Text = "変更後名";
            }
        }
    }
}
```

[Top](#)

## CrystalReportViewerのツールバーに表示されているロゴをカスタマイズしたい

Web用のビューアはロゴの表示/非表示がプロパティで可能ですがWindowsアプリケーション用にはないようです。

次のようにするしか方法はないのかもしれませんが。

以下のコードは適当に書いてみたテストコードです。

要はCrystalReportViewerをコントロールオブジェクトに分解してロゴを表示しているToolStripLabelオブジェクトを取得しています。

あとはそのラベルオブジェクトを煮るなり焼くなり・・・

(ちなみにCrystalReportViewerのツールバーは18個のToolStripItemから構成されていてロゴは18個目です。)

こんなふうになれば色々カスタマイズできるかと思います。

```
private void button1_Click(object sender, EventArgs e)
{
```

```

List lbls = new List();

// コントロールオブジェクトに分解
foreach (Control ctl in crystalReportViewer1.Controls)
{
    // ツールバーを取得
    if (ctl.GetType().Equals(typeof(System.Windows.Forms.ToolStrip)))
    {
        System.Windows.Forms.ToolStrip strip = (System.Windows.Forms.ToolStrip)ctl;
        // ツールバーをアイテムに分解
        foreach (ToolStripItem item in strip.Items)
        {
            // ラベルを取得
            if (item.GetType().Equals(typeof(System.Windows.Forms.ToolStripLabel)))
            {
                lbls.Add(item);
            }
        }
    }
}

Bitmap bmp = new Bitmap("logo.jpg");
lbls[1].BackgroundImage = bmp; // ラベルは2個あってそのうちの2番目がロゴを表示しているラベル
//lbls[1].Visible = false;
}

```

[Top](#)

### CrystalReportViewerのレポートが表示されている部分のイベントをフックしたい。

当然ですが、適切なコントロールにイベントを設定しないと期待した動作は行われません。まずCrystalReportViewerの持つPageViewコントロール(レポートが表示されているところ)を取得します。PageViewはTabControl(「メインレポート」とか表示される部分)を持っているのでこれを取得します。さらにTabControlのが持つDocumentControlを取得します。このDocumentControlがレポートの内容を表示しているそのものです。今回のような場合、こいつに対して設定するとレポートを表示している部分のイベント(マウス動作などなど)をフックすることができます。

```

foreach (Control ctl in crystalReportViewer1.Controls)
{
    CrystalDecisions.Windows.Forms.PageView pv = ctl as CrystalDecisions.Windows.Forms.PageView;
    if (pv != null)
    {
        foreach (Control ctl2 in pv.Controls)
        {
            System.Windows.Forms.TabControl tabCtl = ctl2 as System.Windows.Forms.TabControl;
            if (tabCtl != null)
            {
                foreach (Control ctl3 in tabCtl.Controls)
                {
                    CrystalDecisions.Windows.Forms.DocumentControl dc = ctl3 as CrystalDecisions.Windows.Forms.DocumentControl;
                    if (dc != null)
                    {
                        dc.MouseWheel += new MouseEventHandler(dc_MouseWheel);
                    }
                }
            }
        }
    }
}

```

```
}  
}  
}  
}
```

### 画像を指定して表示したい。

様々な表示方法がありますがここでは三つほど紹介します。

1. クリレポにBlobFieldObjectを貼り付け、プログラムで画像を読み込み、そのObjectに画像データを入れてやる方法。
2. クリレポにOleObjectを貼り付け、適当な式フィールドを用意し、OleObjectの画像保存場所にその式フィールドを指定、  
あとはプログラムからその式フィールドに画像のパスを入れてやる方法。
3. クリレポにOleObjectを貼り付け、画像保存場所に適当な画像ファイルパスを指定しておき、プログラムで  
目的の画像をOleObjectの画像パスに置換してやる方法。

CrystalReportViewerを使う場合は表示を更新しましょう。  
私的には1の方法がお勧めです。

```
//1のサンプル  
System.Drawing.Bitmap bmp = new System.Drawing.Bitmap(@"C:\xxx.jpg");  
using (System.IO.MemoryStream st = new System.IO.MemoryStream())  
{  
    bmp.Save(st, System.Drawing.Imaging.ImageFormat.Png);  
    byte[] bytes = new byte[st.Length];  
    bytes = st.ToArray();  
    row["列名"] = bytes; // rowはDataRowオブジェクト  
}  
  
//2のサンプル  
//rptはReportDocumentオブジェクト  
rpt.DataDefinition.FormulaFields.Item("式フィールド名").Text = @"'C:\xxx.jpg'"   
  
//3のサンプル  
System.IO.File.Copy(@"C:\org.jpg", @"C:\xxx.jpg"); // 画像保存場所に"C:\xxx.jpg"を指定してま
```

## 印刷に関して

### プリンタから直接印刷する。

ReportDocumentオブジェクトのPrintToPrinter()を使うと、CrystalReportViewerコントロールを使わずに直接プリンタから印刷することができます。

```
// C#によるサンプル  
  
// ReportDocumentオブジェクトの宣言
```

```
ReportFile rptObj = new ReportFile();

// ReportDocumentにデータをセット
rptObj.SetDataSource( データセット );

// プリンタの設定
CrystalDecisions.CrystalReports.Engine.PrintOptions printOptions = rptObj.PrintOptions;
// プリンタ名
printOptions.PrinterName = "EPSON      ";
// 用紙サイズ
printOptions.PaperSize = CrystalDecisions.Shared.PaperSize.PaperA4;
// マージン(余白)
CrystalDecisions.Shared.PageMargins margins;
margins.leftMargin = 10;
margins.rightMargin = 10;
margins.topMargin = 20;
margins.bottomMargin = 20;
// マージンの適用
printOptions.ApplyPageMargins(margins);

// 印刷(上記設定を行わない場合、デフォルトのプリンタ及びデフォルトの設定で印刷される)
rptObj.PrintToPrinter(1, false, 0, 0);
```

[Top](#)

### CrystalReportViewerを使って印刷したい。

CrystalReportViewerには印刷ダイアログ表示機能がついているので、プログラム上での設定は必要ありません。

印刷ダイアログでは出力先のプリンタなどを変更できます。

ただし、次の2点に注意が必要です。

- 1 . 出力用紙サイズは変更できますが、レポート自身の大きさをここで変更することはできません。
- 2 . 印刷ダイアログが閉じたときに印刷されたのかキャンセルされたのかを判別することができません。

[Top](#)

### バーコードを出力したい。

バーコードを出力するにはバーコード用のフォントがいります。

CrystalReportsにはバーコードフォントが入っていないので、まずはそのバーコードフォントをインストールします。

バーコードフォントは商用のものからフリーのものまであります、フリーは種類が限られています。

インストールしたら、フィールドの書式設定でフォントをバーコードフォントに変えるだけです。

[Top](#)

### クリスタルレポートの余白をプログラムから設定したい。

次の例は、レポートのページ余白を設定し、プリンタを選択し、レポートを印刷する方法です。

```
[Visual Basic]
Private Sub PrintReport(ByVal printerName As String)
Dim margins As PageMargins
```

```

' PageMargins 構造体を取得し、
' レポートの余白を設定します。
margins = Report.PrintOptions.PageMargins
margins.bottomMargin = 350
margins.leftMargin = 350
margins.rightMargin = 350
margins.topMargin = 350
' ページ余白を適用します。
Report.PrintOptions.ApplyPageMargins(margins)

' プリンタを選択します。
Report.PrintOptions.PrinterName = printerName

' レポートを印刷します。startPageN および endPageN
' パラメータを 0 に設定し、すべてのページを印刷します。
Report.PrintToPrinter(1, False, 0, 0)
End Sub

```

```

[C#]
private void PrintReport(string printerName)
{
PageMargins margins;

// PageMargins 構造体を取得し、
// レポートの余白を設定します。
margins = Report.PrintOptions.PageMargins;
margins.bottomMargin = 350;
margins.leftMargin = 350;
margins.rightMargin = 350;
margins.topMargin = 350;
// ページ余白を適用します。
Report.PrintOptions.ApplyPageMargins(margins);

// プリンタを選択します。
Report.PrintOptions.PrinterName = printerName;

// レポートを印刷します。startPageN および endPageN
// パラメータを 0 に設定し、すべてのページを印刷します。
Report.PrintToPrinter(1, false,0,0);
}

```

```

[C++]
void PrintReport(String* printerName)
{
PageMargins margins;

// PageMargins 構造体を取得し、
// レポートの余白を設定します。
margins = Report->PrintOptions->PageMargins;
margins.bottomMargin = 350;
margins.leftMargin = 350;
margins.rightMargin = 350;
margins.topMargin = 350;
// ページ余白を適用します。
Report->PrintOptions->ApplyPageMargins(margins);

// プリンタを選択します。
Report->PrintOptions->PrinterName = printerName;

// レポートを印刷します。startPageN および endPageN

```

```
// パラメータを 0 に設定し、すべてのページを印刷します。  
Report->PrintToPrinter(1, false,0,0);  
};
```

[Top](#)

## 外字印刷

外字が化けてしまう場合は以下のサイトよりサービスパックを当ててみます。

[http://support.businessobjects.com/downloads/updates/service\\_packs/crystal\\_reports\\_jp.asp#Crystal\\_Reports](http://support.businessobjects.com/downloads/updates/service_packs/crystal_reports_jp.asp#Crystal_Reports)

[Top](#)

## その他

### 製品版とバンドル版の違い

VS.NET Professional以上にはクリスタルレポートがバンドルされています(Standardにはついてない)。  
このバンドル版CrystalReportsは製品版と比べて、以下のような機能制限があります。

- \* データ接続先の制限
- \* エクスポート形式の制限 ( CSV, XML 無し )
- \* ガイドライン/垂直ルーラ無し
- \* レポートアラート無し
- \* ActiveX ビューア、Java ビューア無し
- \* Crystal Reports Server の利用不可
- \* その他

また、製品版はデザイナーでレポートをプレビューできるため、レポートを迅速にかつ正確に作成できます。  
プレビュー側からデザインを編集できるのも良いと思います。

これらの制限はデザイン時の効率に影響するものばかりなので、機能的には十分実用可能です。

あとはCrystalReportsのグレードによっていろいろと機能が増えていきます(詳しくは製品の紹介を参照)。

[Top](#)

### VS.NETで実行ファイルにレポートファイル(rptファイル)を埋め込む

CrystalReportsで作成したレポートファイルはプロジェクトに追加できます。

追加はメニュー[ファイル]から[既存項目の追加]を選ぶことでできますが、実行ファイルに埋め込むにはどうするのでしょうか。

これはソリューションエクスプローラで追加されたレポートファイルのプロパティを開き、ビルドアクションを[埋め込まれたリソース]にすればOKです。  
これにより、アプリケーションを配布するときにレポートファイルが必要なくなります。

[Top](#)

## レポートファイルは実行ファイルに埋め込んだ方が良い？

---

レポートファイルを埋め込むか否かによって、配布の方式が違ってきます。埋め込むと配布するのは実行ファイルだけで済みますが、埋め込まない場合はレポートファイルの配布も必要になります。

どちらが便利かは簡単には判断できません。これはアプリケーションがどれだけレポートファイルに依存するかに関わってきます。

具体的に言うとADO.netのDataSetを使う場合は、レポートファイルの配布はほとんど意味をなさないでしょう。なぜならレポートファイルに用いるデータベーススキーマはDataSetのスキーマであり、そのDataSetはプログラム内で作られるからです。つまり、プログラムの修正とレポートが密接に関わってくるため、どちらかのみ修正というケースが少ないはずで。

逆にDataSetを用いない場合、極端な例でいくとプログラム側はレポートファイルに印刷命令を出すだけで、どんなデータをどのように表示するかを全部レポートファイルにまかせる場合に、実行ファイルとレポートファイルを分けることのメリットが生まれることでしょう。

そして、CrystalReportsはSQL、クリスタル構文(BASIC構文)を用いてプログラム並みの作業をこなすことができ、実はビジネスロジックのほとんどをCrystalReportsで実装できます。

しかし、大抵のプログラマはプログラム側でデータを作成、編集した方が効率が良いと考えていると思うのでレポートファイルはほとんどデータ印刷用と化しているように思います。

[Top](#)

## Basic構文とCrystal構文の違い。

---

CrystalReportsで式を作成する場合、Basic構文で記述するかCrystal構文で記述するかを決めなければなりません。

レポート中でこれらを混用することはできますが、一つの式の中では混用できません。

どちらを使っても全く問題なく、やれることは大概いっしょです。

Basicに慣れている人ならBasic構文の方がやりやすいかもしれません。

ただし、サンプル例として挙げられるのはCrystal構文で書かれた式の方が多いです。

[Top](#)

## CrystalReportsのバージョンアップする際、前の環境を残しておきたい。

---

CrystalReports9からはサイドバイサイドでインストールできます。

つまり一つのPCに違うバージョンのCrystalReportsを共存させることができます。

[Top](#)

## CrystalReportsを用いたアプリケーションの配布について。

---

[ここ参照。](#)

[Top](#)

## TableModuleとTableDataGateway。

---

[ここ参照。](#)

[Top](#)

## CrystalReportsの関数のヘルプはないの？

---

バージョンは少々古いですが、以下のサイトより関数ヘルプを落とせます。

[http://support.businessobjects.com/communityCS/FilesAndUpdates/cr8\\_formularef\\_japanese.zip.asp](http://support.businessobjects.com/communityCS/FilesAndUpdates/cr8_formularef_japanese.zip.asp)

[Top](#)

---

## CrystalReportsで印刷時に文字化けする。

---

.NET Frameworkとプリンタドライバの相性の問題の可能性が高いと思われます。  
プリンタドライバを最新のものにしてみることをお勧めします。

[Top](#)

---

## "~"の文字化け。

---

これは、Oracle9.0.1.4.0以上、Oracle9.2以上とWindowsとで、SJISとUnicode間の変換を行う時の、Unicodeの対応が異なる為に発生するらしいです。直し方はデータベースの文字コード体系の設定を変えればいいらしいですが、いきなりデータベースの設定を変えるわけにもいかず試せてないので正直どうしたらいいかわかりません。

[Top](#)

---

## 「このフィールドは集計できません」というエラーの意味。

---

このエラーは非循環フィールドをSum()で集計しようとしたときに発生します。  
Sum()は循環フィールドを集計する関数です。

非循環フィールドの例には以下のようなものがあります。

- ・テキストフィールド
- ・変数を用いた式フィールド
- ・評価時期変更関数を用いた式フィールド
- ・積算合計フィールド(Sum())を用いて集計したものも含む)
- ・その他

このエラーを回避するにはSum()を使わずに変数によって集計するようにします。

```
//グループヘッダーに以下の式フィールドを挿入
@合計のリセット//非表示にしておく。
WhilePrintingRecords;
numberVar TotalValue:=0;

//詳細に以下の式フィールドを挿入
@合計のカウント//非表示にしておく。
WhilePrintingRecords;
numberVar TotalValue;
TotalValue := TotalValue +{非循環フィールド}

//グループフッターに以下の式フィールドを挿入
@合計の出力
WhilePrintingRecords;
numberVar TotalValue;
```

[Top](#)

---

## CrystalReportsをバージョンアップしたら罫線の長さが太くなった。

---

これは例として、CrystalReports4.6で作成したレポートをCrystalReports10で開いたときに起こることが報告されています。細線で作成したはずの罫線が項目ブレイク毎に太くなるというものです。

CrystalReportsはバージョンアップに際し、下位互換性を保つためにサービスパックが提供されています。なるべくそのバージョンのサービスパックを適用しておくのが無難だと思われます。

[Top](#)

---

## データベース構造の変更をレポートに反映させたい。

---

メニュー[データベース]から[データベースの照合]を実行します。

これによって、データベース構造の再取得を行い、フィールドを再構成します。保存されていたデータベースのデータも最新のものに置き換わります。

保存されていたデータベースのデータだけ最新のものにしたい場合は、メニュー[レポート]から[レポートデータの最新表示]を実行します。

しかし、デフォルトではメニュー[ファイル]の[レポートオプション]の[最初の最新表示時に照合]にチェックが入っているため、[レポートデータの最新表示]を実行すれば、同時に[データベースの照合]も行われます(つまりデフォルトでは機能が同じ)。

[Top](#)

---

## 特殊フィールドのレポートタイトルとかレポートコメントって何？

---

特殊フィールドの一つであるレポートタイトルまたはレポートコメントは、その名の通りレポートのタイトルやコメントを表示するフィールドですが、レポートに埋め込むことだけでは使うことができません。

この値は[ファイル]メニューの[プロパティ]から「ドキュメント プロパティ」を開き、[概要]タブで指定することによって変更が可能です。

[Top](#)

---

## CrystalReportViewerのツールチップを抑制する。

---

CrystalReportViewerでオブジェクトにカーソルを当てるとツールチップテキストが表示されます。これを表示しないようにするには、以下のように設定します。

1. デザインで表示させたくないフィールドの書式設定を開きます。
2. [共通]タブの「ツールヒント テキスト」の条件付き書式設定を開きます。  
(右にある[x+2]みたいなボタンを押し、式エディタを開く)

3. 以下の式を挿入します。  
WhilePrintingRecords;  
Chr(9);

[Top](#)

---

### WhitePrintingRecords

WhitePrintingRecordsとは条件式を評価するタイミングを指定する宣言文です。  
通常、条件式は式の中でデータベースのデータを利用するかしないかで評価のタイミングが異なります。

[利用している場合] データを読み込む時に評価

[利用していない場合] データを読み込む前に評価

しかし、これを条件式の前で宣言しておく、その条件式はデータをレポートに出力している最中に評価されるようになります。

例えば、積算合計などを変数を使って表示する場合、これを宣言しておかないと、まだ何も代入されていない合計値(つまり0)が出力されてしまうことになります。

なので、ちゃんと値が計算結果が代入されてから表示しましょうという意味を込めて

WhilePrintingRecordsを宣言しておきます。

WhilePrintingRecordsを使うならその条件式に関連する条件式全てに、タイミングを合わせるために一応WhilePrintingRecordsを宣言しておくべきでしょう。

サンプルに示すのは変数を用いた積算合計の算出例です。

しかし、このような単純な積算合計ならば、デザイナーに用意されている積算合計フィールドを使うのが一番てっとり早いと思います。

```
//グループごとの積算合計を表示したい場合、以下のようにする。
```

```
//グループヘッダーに以下の式フィールドを挿入。
```

```
WhilePrintingRecords;
```

```
CurrencyVar Amount := 0;
```

```
//詳細に以下の式フィールドを挿入。
```

```
WhilePrintingRecords;
```

```
CurrencyVar Amount;
```

```
Amount := Amount +{価格}
```

```
//グループフッターに以下の式フィールドを挿入
```

```
WhilePrintingRecords;
```

```
CurrencyVar Amount;
```

[Top](#)

### CurrencyVar

CrystalReportsの構文で用いる変数の型です。

CurrencyVarは通貨型とも言うべきでしょうか。

この型で宣言した変数は通貨として扱われ、通貨記号(\マーク)がつきます。

ただし、通貨型といえど計算は通常の数値と同様に行われるので、数値型の代用として用いられることが多いです。

ちなみに数値型はNumberVar、文字列型はStringVar、日付型はDateTimeVarになります(他にもいろいろあるけどヘルプ参照のこと)。

## ドリルダウン

ドリルダウンとは、現在表示されているデータから一段階下の詳細なデータに掘り下げる行為全般を指します。

例えば、サブレポートを挿入してそいつをメインから参照する行為や、そのグループのみに着目する行為などは全てドリルダウンしていることになります。

CrystalReportsでドリルダウンを禁止することはできません。なぜならドリルダウンはアプリケーションの機能というより人間の行為に近く、またその言葉の指す範囲が広いからです。

しかし、CrystalReportsにはドリルダウンに関する機能がたくさんあります。中でもよく使われるのにやっかいなのが、セクションエキスパートで設定できる次の二つです。

- ・「非表示-ドリルダウン可」
- ・「非表示-ドリルダウン不可」

可とか不可とか書いてありますがドリルダウンを禁止するという意味ではありません。そのセクションを非表示にした際、ドリルダウン先のセクションの内容を表示するかしないかの違いです。

ゆえに「非表示-ドリルダウン不可」はドリルダウンする意味がないので禁止に一番近いと思われま

す。

## 条件付き書式設定

CrystalReportsの設定項目には各所に「 $\times \cdot 2$ 」と鉛筆マークみたい絵柄のボタンが見受けられると思います。

これが条件付き書式設定を行うボタンです。

ボタンを押すと式を作成するエディタが開くのでここに条件式を入力します。

条件が評価されるタイミングで、評価結果より動的にその項目の設定を変更できます。

例えば、フィールドの書式設定の共通タブにある「非表示」項目の条件付き書式設定を指定すると評価結果が真の場合に「非表示」にチェックが入ります(つまり非表示になる)。

チェックボックスの操作だけでなく、フォントやフォントのサイズの切り替えなども動的に行うことができます。

このような様々な動作を規定できるため、条件式の戻り値はその項目に依存します。

## Shared変数

CrystalReportsで使う専用のスクリプト言語(Crystal構文)で用いる変数のスコープです。変数のスコープには以下のような三種類が用意されています。

- ・ Global (グローバル変数)  
宣言したレポートで変数が共有されます。

- ・ Shared (共有変数)  
現在のレポートおよびサブレポートで変数が共有されます。  
最も効果範囲の広いスコープです。

- ・ Local : (ローカル変数)  
変数が定義されている式の中でのみ共有されます。

変数に何も指定しなければGlobalとして扱われます。

```
// ページヘッダーに次のような式フィールドを挿入する。
@初期化          // 式フィールド名
numberVar count; // Globalな数値型変数countの使用を宣言する。
count := 0;      // countに0を代入する。

// 詳細に次のような式フィールドを挿入する。
@インクリメント
numberVar count; // Globalな数値型変数countの使用を宣言する。
//このcountは@初期化で宣言しているcountと同一のものであり、この時点ですでにcountには0が入っている。
count := count + 1; //countに0+1(=1)を代入する。

//詳細があるたびにcountは1ずつ増え、ページヘッダーでリセットされる。
//よって、countの役割は1ページ内のレコード数をカウントすることである。
```

[Top](#)

## 続くセクションをアンダーレイ

CrystalReportsは機能の名称から機能の動作を連想しにくい事例が多々あります。  
この機能は主に図などを利用する場合に用いられるのではないのでしょうか。

これをチェックするとセクションの内容を、出力するときに次のセクションにマージしちゃいます。  
例えば以下のような例でグループヘッダーのこの設定にチェックを入れたらどんな風になるでしょうか。

-----グループヘッダー-----

会社名

-----詳細-----

A会社      社長

(プレビュー)

・チェック入れなかった場合 ・チェック入れた場合

会社名 会社名 A会社      社長

A会社      社長

デザイン時にちゃんとマージされることを考慮しとかなないと重なってぐちゃぐちゃになってしまいます。

[Top](#)

## データプロバイダ

.NET Frameworkからデータベースにアクセスするために最初に行うのは、このデータプロバイダの選択です。

データプロバイダとはアプリケーションとデータソース(データベースやXMLなど)の橋渡しを行うものです。

そして、データソースの種類によりデータプロバイダにも様々な種類が存在します。

[Top](#)

## レポートファイル

---

CrystalReportsで作成する拡張子rptのファイルのことです。  
.NET Frameworkではレポートファイルをプロジェクトに追加すると同時に、ビルド時に、リソースとして埋め込まれるように設定され、なおかつプログラム内で扱えるようにオブジェクトにマージしてくれます(レポートファイルのクラスができる)。

[Top](#)

---

## 積算合計フィールド

---

グループ化するときには総和やレコード数などを計算して埋め込むフィールド。

[Top](#)

---

## 特殊フィールド

---

出力日付やページ番号などのフィールド。

[Top](#)

---

## 式フィールド

---

実行時にCrystalReportsの関数を使って計算した式を埋め込むためのフィールド

[Top](#)

---

## パラメータフィールド

---

レポートに渡す変数となる値です。  
パラメータフィールドには、プログラムから値を設定できます。  
また、条件に合致するフィールドだけを抽出するときにも使われます。

[Top](#)

---

## グループ名フィールド

---

グループ化するときのフィールドです。どの項目でグループ化するかを定義します。  
グループ名フィールドを追加すると、「グループヘッダー」と「グループフッター」も追加されます。

[Top](#)

---

## SQL 式フィールド

---

接続先のデータソースがデータベースの場合、フィールドエクスプローラに「SQL 式フィールド」の項目が現れます。  
データベース以外(DataSetやExcel)ではSQL 式フィールドは現れず、使うことはできません。

このフィールドは扱的には式フィールドといっしょで、式の中で演算をしたり評価したりするのに使います。

式フィールドとSQL 式フィールドではその演算の仕組みとタイミングが異なります。  
SQL 式フィールドにおける演算はSQLを用いて全てサーバー側で行われます。よって使う構文はSQLであり、演算の仕様はSQLに、その仕組みはデータベースに依存します。

式フィールドはCrystalReports側で演算を行うので、メモリ上にデータを展開して行うため、サーバー側で行うのに比べてスピードが遅く、マシンスペックに依存するので安定性に欠けることとなります。

ゆえにスピードと安定性を求めるなら式フィールドではなくSQL 式フィールドを使う方が好ましいですが、機能的には式フィールドの方が優れているので、まあ使い分けが必要なわけですし、レポート単位でどちらを使用するか決めた方が予期せぬバグが起こらなくて良いと思います。

[Top](#)

---

## License Key

---

CrystalReportsで作成したレポートファイルを含んだアプリケーションをクライアントで動作させるには・・・結構めんどくさいです。  
とりあえずまず初めに必要なのがこのLicense Key(登録キーコード)と呼ばれるキーコード。これは開発もとのBusiness Objectsで登録するともらえるようです。登録にお金はかからないので、どうやらただみたいです。

製品版を購入すると製品アクティベーションキーコード(俗に言うCD-Keyとかシリアルナンバーみたいなもの)が付属してきますが、これをLicense Keyの代わりに使うこともできます。

[Top](#)

---

## 循環フィールド

---

クリスタルレポートはレコード単位でデータを参照する仕組みを持っています。  
通常、デザインで配置されるフィールドはデータベースのフィールド(Column)に相当し、レコードが遷移することによって表示するデータ(情報)が変化します。  
このようなフィールドは表示する情報がデータベースに完全に依存し、かつレコード単位で遷移することから循環フィールドと呼ばれることがあります。

[Top](#)

---

## 非循環フィールド

---

循環フィールドでないものを非循環フィールドと呼ぶことがあります。  
つまり非循環フィールドとは、データベースに依存しないフィールドのことです。

例えば以下のようなフィールドが非循環フィールドに当たります。

- ・テキストフィールド
- ・変数を用いた式フィールド
- ・評価時期変更関数を用いた式フィールド
- ・積算合計フィールド(Sum())を用いて集計したものも含む
- ・その他

{値段}という循環フィールドがあったとします。

例1.

```
//@値段1  
{値段} * 1.05;
```

//@値段1は循環フィールド

例2.

```
//@値段2  
numbarVar zei;  
zei := 0.05;  
{値段} + {値段} * zei
```

//@値段2は非循環フィールド

例3.

```
//@ 値段3
```

```
WhilePrintingRecords;
```

```
{値段} * 1.05;
```

```
//@ 値段3は非循環フィールド
```

[Top](#)

---

## Chr(13) & Chr(10)

---

Chr(13) & Chr(10)

は復帰改行文字を表します。

Chr(13)は復帰文字"\r"、

Chr(10)は改行文字"\n"

を表す。

Windowsの改行は通常、復帰改行文字です。

[Top](#)

---

## ADO.NETのよくあるエラーと解決方法

---

### 追加情報 : Parameter 'PARAM1': No size set for variable length data type: String.

---

[意味]

パラメータ'PARAM1'でエラー発生。可変長さデータ・タイプ (string型) にセットできるサイズが存在しません。

[説明]

パラメータ変数にNULLを代入したときに発生する。

SQL文のパラメータ(OracleParameter)はデフォルトでNULLの代入を許さないようになっている。

NULLを許可させるためには、パラメータオブジェクトのIsNullableプロパティをTrueにする。

OracleDataAdapterを利用している場合、IsNullableをセットすることは不可能であると思われる。

```
C# OracleParameterクラスを使った例
```

```
string str = "パラメータデータ";
```

```
System.Data.OracleClient.OracleCommand command = new System.Data.OracleClient.OracleCommand();
```

```
// 変数名"PARAM1"、データ型"VARCHAR"、サイズ"20"でOracleParameterオブジェクト"parameter"を宣言
```

```
System.Data.OracleClient.OracleParameter parameter = new System.Data.OracleClient.OracleParameter("PARAM1",
```

```
parameter.IsNullable = true;
```

```
parameter.Value = str;
```

```
command.Parameters.Add( parameter );
```

[Top](#)

---

### 追加情報 : ORA-01400: cannot insert NULL into ("PESON\_DB"."PESON"."ADDRESS")

---

[意味]

"ADDRESS"フィールドにNULLを挿入できません。

[説明]

"ADDRESS"フィールドに"NOT NULL"制約が存在する。そこにNULLを代入しようとしたために発生する。

SQL Plusで"NOT NULL"制約を解除すれば良い。

```
// "NOT NULL"制約を解除する場合
SQL> alter table PERSON modify ADDRESS varchar(50) null;
// "NOT NULL"制約を付加する場合
SQL> alter table PERSON modify ADDRESS varchar(50) not null;
//commitを忘れずに
```

[Top](#)

---

### 追加情報 : ORA-01008: not all variables bound

---

#### [意味]

一部パラメータが存在しません。

#### [説明]

パラメータつきクエリでパラメータが宣言されていないときに発生する。  
OracleCommand.CommandTextに設定しているクエリのパラメータと  
OracleCommand.Parametersに設定しているパラメータを確認する。

[Top](#)

---

### 追加情報 : ORA-01036: 変数の名前/数が無効です。

---

#### [意味]

パラメータの名前もしくは数が一致しない。

#### [説明]

パラメータの名前もしくは数が一致しない場合に発生する。  
クエリのパラメータとOracleCommandに設定したパラメータの数およびスペルを確認する。  
よくある間違いを以下にあげる。

- ・クエリのパラメータの順番が違う。
- ・":"の間違い。クエリ中でパラメータ名には":"をつけるが、":"を省いた部分がパラメータ名であるのでOracleParameterでは":"を省いて指定する。
- ・大文字小文字の間違い。
- ・単純にパラメータが抜けている。

[Top](#)

---

### 追加情報 : No data exists for the row or column.

---

#### [意味]

行または列にデータが存在しません。

#### [説明]

データを取得しようとしたときに、データがNULLだと発生する。  
IsDBNull()メソッドでNULLかどうか調べてから取得するようにする。

[Top](#)

---

### 追加情報 : ORA-00920: invalid relational operator

---

#### [意味]

比較演算子が無効です。

#### [説明]

比較演算する箇所で比較演算が不正・できないときに発生する。

'=' , '<' , '>' , '<=' , '>=' , '<' , '>'などの比較演算子が正しく記述されているか確認する。

[Top](#)

---

### 追加情報 : ORA-00923: FROM keyword not found where expected

---

[意味]

FROMキーワードが指定位置に見つかりません。

[説明]

SQLクエリにFROMキーワードがない、もしくは','などの忘れによって認識できない場合に発生する。SQLクエリを良く見直せば案外簡単に直るエラー。

[Top](#)

---

### 追加情報 : ORA-01008: バインドされていない変数があります。

---

[意味]

OracleParameterがセットされていない。

[説明]

SQLクエリでパラメータ変数を使用しているにもかかわらず、OracleParameterがOracleCommandにセットされていないときに生じる。

大抵、パラメータ文字列の不一致により起きる。

```
//C# OracleParameterクラスを使った例
```

```
string str = "パラメータデータ";
```

```
System.Data.OracleClient.OracleCommand command = new System.Data.OracleClient.OracleCommand();
```

```
// 変数名"PARAM1"、データ型"VARCHAR"、サイズ"20"でOracleParameterオブジェクト"parameter"を宣言
```

```
System.Data.OracleClient.OracleParameter parameter = new System.Data.OracleClient.OracleParameter("PARAM1",
```

```
parameter.IsNullable = true;
```

```
parameter.Value = str;
```

```
command.Parameters.Add( parameter );
```

[Top](#)

---

### 追加情報 : ORA-00936: 式がありません。

---

[意味]

SQLクエリに必要な式がない。

[説明]

SQLクエリのWhere句に、式がないまたは式が認識できない場合に起きることがある。

[Top](#)

---

### 追加情報 : ORA-00918: 列の定義が未確定です。

---

[意味]

データベースにSQL文に書いた列が存在しない。

[説明]

単純に列名を間違えているとか、複数テーブルなどからデータを取得している場合にはどのテーブルの列かを指定する必要があるのに、それがいないためにどのテーブルの列か認識できない場合に起きることがある。

[Top](#)

---

## 追加情報 : ORA-00933: SQL command not properly ended

---

### [意味]

SQLコマンドが正しく終了されていない。

### [説明]

SQLクエリが正しくない場合に起きる。様々な理由が考えられるが「,」などの忘れがないかを確認する。

[Top](#)

## リンク集

### クリスタルレポート関連

---

[むかむか クリスタルレポート入門](#)

[Top](#)

---