TEKNISA Training

BIRT

Business Intelligence and Reporting Tools

NOV 2025

BIRTV01.1

Conteúdo

Introdução

O **BIRT** (*Business Intelligence and Reporting Tools*: Inteligência de negócio e ferramenta de relatórios) é uma ferramenta que fornece relatórios para aplicações *runtime* no cliente e aplicações *web*, especialmente as aplicações baseadas em *Java*.

O BIRT trata de uma ampla gama de necessidades de relatórios fornecendo recursos de relatórios essenciais, como *layout* do relatório, acesso a dados e *script*.

Além do BIRT, um visualizador/renderizador PDF para aplicações *web* embutido no *Apache Tomcat Server* que pode funcionar paralelamente com um servidor *web*/PHP.

Para mais informações, acesse `http://jakarta.apache.org/tomcat/`.

Este manual tem como objetivo apresentar alguns dos **componentes, funcionalidades e convenções de uso do BIRT**.

Documento produzido e organizado por Rafael Pick Brenner.

Componentes do BIRT

Esta seção detalha as características e capacidades dos **componentes do BIRT**:

2.1 *LABEL*

Componente utilizado para exibir textos fixos. Seu uso mais comum é em cabeçalhos de tabelas e de relatórios.

2.2 *TEXT*

Exibe um texto de uma ou mais linhas, previamente definido. Pode incluir formatação HTML e valores computados.

2.3 *DYNAMIC TEXT*

Exibe o conteúdo de um campo de banco de dados que pode conter formatação via código HTML. Neste componente, podem-se realizar manipulações via linguagem JavaScript.

2.4 *DATA*

Exibe o conteúdo de um campo de banco de dados ou um valor computado. Aceita formatações, inclusive condicionais. Os valores exibidos podem ser manipulados via JavaScript, e seu uso mais comum é em tabelas.

2.5 *IMAGE*

Exibe qualquer tipo de imagem suportada por um navegador (*web browser*), de forma estática ou dinâmica. Suporta imagens a partir de uma URL, de um banco de dados ou de uma pasta.

2.6 *GRID*

Fornece um arranjo tabular de componentes, funcionando como um contêiner que os posiciona de forma lógica e prática. Este componente é similar a uma tabela HTML.

2.7 *LIST*

Apresenta os dados de um *dataset* em forma de lista. Por padrão, possui seções de cabeçalho, detalhe e rodapé. Os componentes na seção de detalhe são repetidos para cada registro do *dataset*, enquanto o cabeçalho e o rodapé são exibidos uma única vez, no início e no fim da lista, respectivamente.

2.8 *TABLE*

Funciona de forma similar ao componente *List*, com a principal diferença de que exibe os dados em formato tabular, podendo conter uma ou várias colunas.

2.9 *CHART*

Exibe gráficos em diversos formatos, como colunas, linhas, pizza, entre outros.

2.10 *CROSS TABLE*

Tem a função de exibir dados de um *data cube*. Seu funcionamento é similar ao de uma *Table*.

2.11 *AGGREGATION*

Fornece funções de agregação, tais como SUM, AVERAGE, COUNT, MIN e MAX.

Princípios Básicos

Esta seção tem o objetivo de informar o **princípio básico de funcionamento de alguns componentes do BIRT**. Características mais detalhadas serão aprofundadas no decorrer deste manual.

3.1 *DATA SET*

O *data set* é o receptáculo da *query*, comportando apenas uma *query* por vez. Quaisquer outros comandos não envolvidos diretamente com a consulta de dados (ex: INSERT, UPDATE, DESCRIBE) não são suportados.

O *dataset* associa os parâmetros do relatório com os parâmetros da *query*. Adicionalmente, permite que o resultado da consulta seja reordenado e filtrado, e possibilita a alteração do texto da *query* via *script*.

3.2 *TABLE*

Este componente *Table* é responsável por listar os resultados dos *data sets*. Permite filtrar, substituir valores, ordenar, agrupar e aplicar formatação condicional aos resultados.

A tabela possui três seções distintas: cabeçalho, detalhe e rodapé.

1. O cabeçalho apresenta os dados da tabela toda vez que a tabela é iniciada e é configurado para se repetir a cada nova página.
2. A seção de detalhe exibe os resultados do *dataset*, organizando cada registro em uma ou mais linhas, conforme a configuração. Além disso, retoma a exibição de acordo com as quebras de páginas.
3. O rodapé é semelhante ao cabeçalho. Entretanto, é exibido apenas uma vez, no final da tabela.

É importante notar que, se o cabeçalho ou o rodapé contiverem campos de dados (*Data*) semelhantes, estes exibirão a informação correspondente ao primeiro registro do *dataset*.

A tabela também oferece a funcionalidade de agrupamento de dados. Através dela, pode-se definir um campo para agrupar os resultados, o que cria seções de "sub-cabeçalho"e "sub-rodapé"entre os dados exibidos na área detalhes.

O processo agrupa todos os registros com o mesmo valor, exibe o sub-cabeçalho, lista os registros e, por fim, exibe o sub-rodapé, repetindo o ciclo para o próximo grupo. Múltiplos níveis de agrupamento são permitidos, sendo que o primeiro nível adicionado será o mais abrangente.

3.3 *GRID*

O componente *Grid* apresenta um comportamento semelhante a uma *table*. Entretanto, não possui cabeçalho e rodapé, funciona como a área de detalhes da *table*, e não tem a capacidade de agrupamento.

Herança de Formatação

Um relatório apresenta uma estrutura aninhada, com componentes contidos dentro de outros. Por exemplo, um componente *Grid* de duas colunas pode conter uma linha composta por duas *tables* (em cada uma de suas células). Por sua vez, cada *Table* pode possuir componentes de *Data* em sua seção de detalhe e componentes *Label* em seu cabeçalho.

Alterar a formatação de cada componente individualmente seria uma tarefa trabalhosa. Com o conceito de **herança de formatação**, o processo é simplificado. Ao alterar a fonte do grid, as alterações são propagadas para os componentes internos (como as *Tabelas* e os campos de *Data*).

Na ilustração a seguir há um exemplo dos subcomponentes de uma *grid* e seus níveis hierárquicos. O **Grid** é o contêiner geral, que contém *Rows*, que por sua vez contêm *Cells*. A Célula é o contêiner final que pode abrigar outros componentes, como *Labels*, *Tabelas* ou até mesmo outro *Grid*. Esta mesma lógica se aplica a uma *Table*.

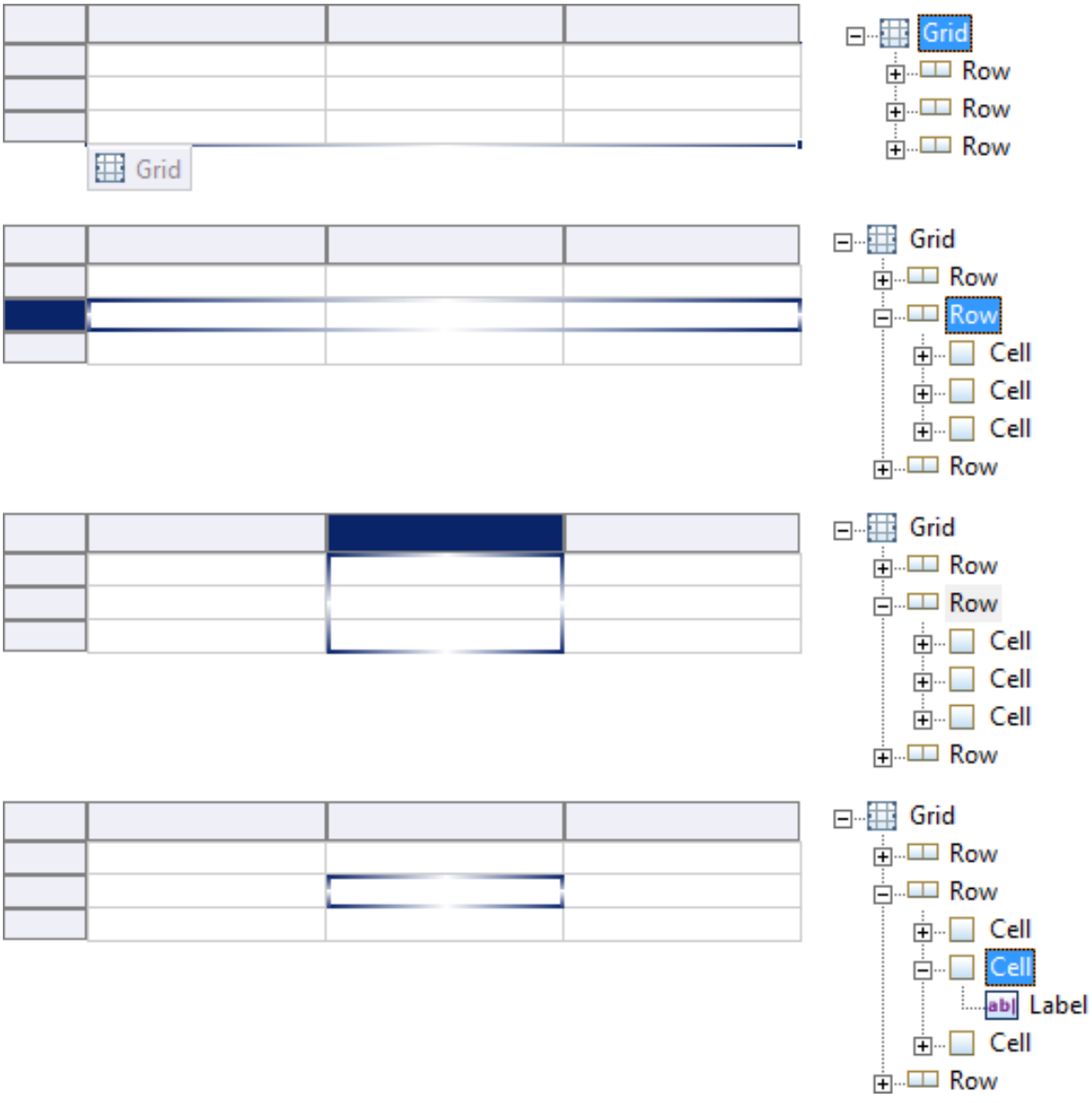


Figura 1

Convencionamos: no exemplo do *grid* o nível inferior seria o *grid*, seguido pelo *row*, até o nível superior mais alto do *grid* (a *cell*).

Quando se realiza uma alteração na formatação de um nível, os níveis superiores e seus componentes contidos são afetados. No entanto, formatações vindas de níveis inferiores não serão propagadas para um nível superior se este já possuir uma formatação própria.

Em resumo: **a formatação de níveis superiores tem preferência sobre a de níveis inferiores.**

Property Editor

A aba **Property Editor** contém os painéis mais importantes usados no BIRT. Nesta aba são encontradas sub-abas e painéis de configurações de todos os componentes do BIRT. Os painéis são exibidos conforme o componente selecionado.

Estas propriedades ficam gravadas em um XML embutido no arquivo do relatório e é acessível ao usuário, tanto para visualização quanto para edição direta.

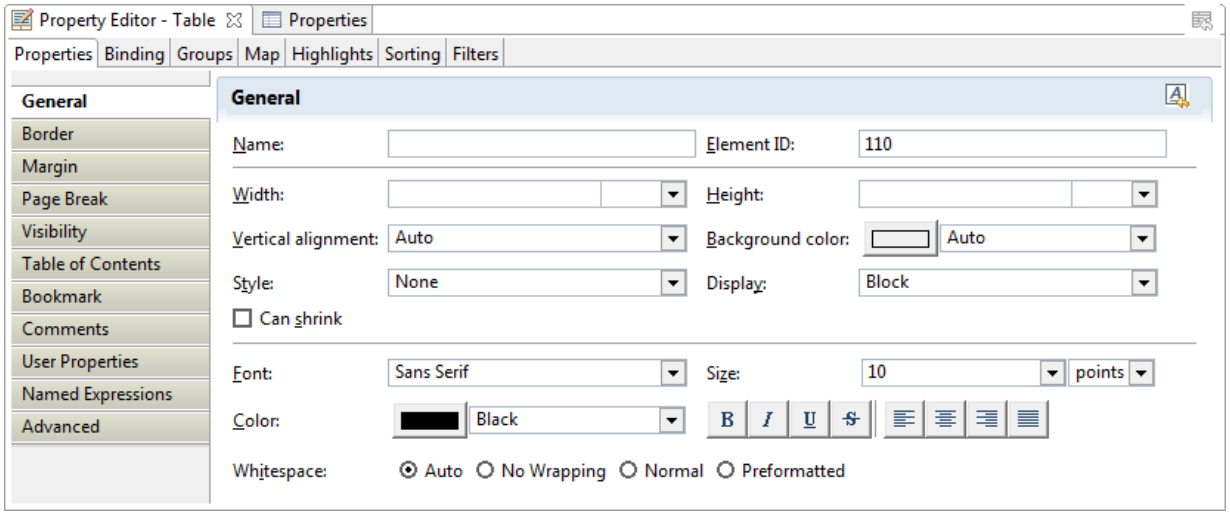


Figura 2

5.1 PROPERTIES

Properties é a primeira aba do *Property Editor* e é a que contém mais painéis de configurações.

A seguir os painéis serão detalhados:

5.1.1 *General*

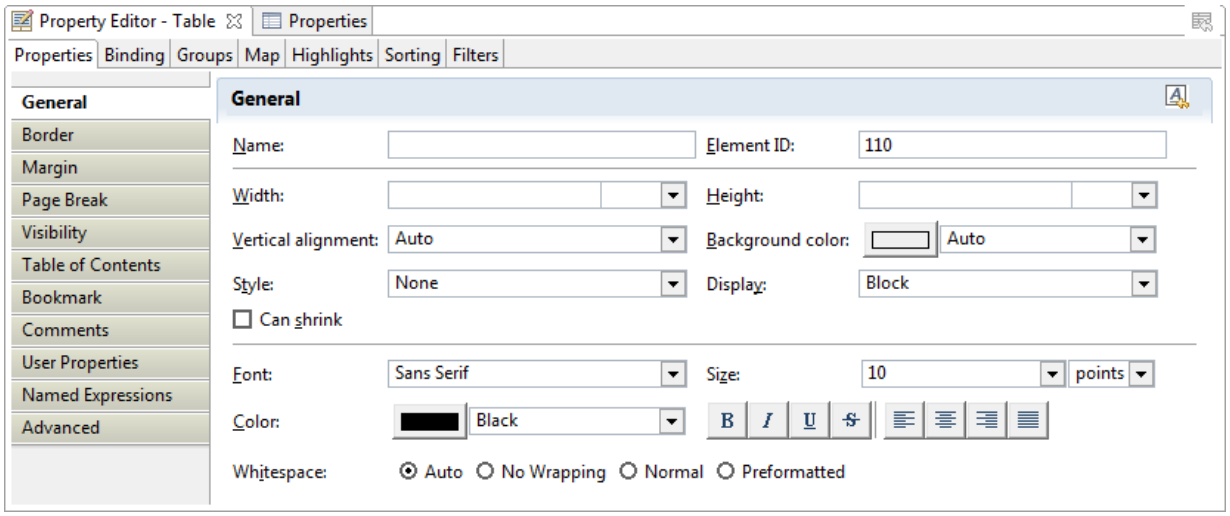


Figura 3

A seleção da opção **"Auto"** para uma propriedade faz com que o componente herde a característica correspondente de seu hospedeiro.

- **Width e Height:** Definem a largura e a altura do componente. Se não informados, o componente se ajustará à maior largura possível dentro de seu contêiner e à menor altura necessária para seu conteúdo.
- **Vertical alignment:** Define o alinhamento vertical do componente. Esta propriedade tem efeito quando o componente não ocupa todo o espaço vertical de seu contêiner.
- **Can shrink:** Quando habilitada, esta opção ajusta (encolhe) o componente ao tamanho de seu conteúdo. Só funciona se as propriedades `Width` e `Height` não forem definidas.
- **Display:**
  - **Inline:** O componente ocupa apenas o espaço necessário para seu conteúdo na linha.
  - **Block:** O componente ocupa verticalmente o espaço de seu conteúdo e não necessariamente provoca uma quebra de linha.
- **Whitespace:** Define o comportamento da quebra de linha em textos.
  - **No Wrapping:** Impede a quebra de linha, mantendo o texto em uma única linha, independentemente do espaço disponível.
  - **Normal:** Permite a quebra de linha automática em espaços em branco quando o texto excede o espaço disponível.
  - **Preformatted:** Preserva todos os espaços em branco formatados, demais espaços poderão sofrer quebra de linha.

5.1.2 *Border*

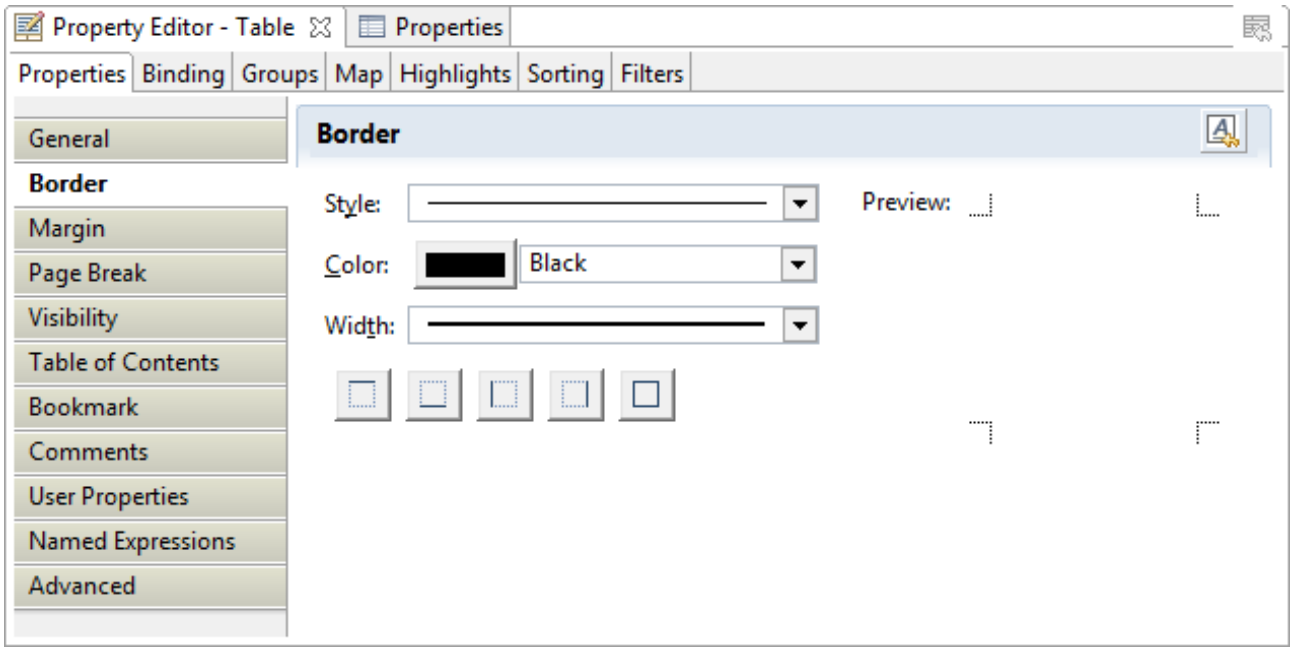


Figura 4

A propriedade de borda não utiliza herança. Cada elemento de um *Grid*, *Table*, *Cross Table* ou *List* possui suas próprias configurações de borda; ou seja, o componente, a linha (*row*) e a célula (*cell*) têm bordas independentes.

Se, por exemplo, uma borda for aplicada a uma *row* de um *Grid*, esta **não** será propagada para as *cells* desta *row*.

5.1.3 *Margin & Padding*

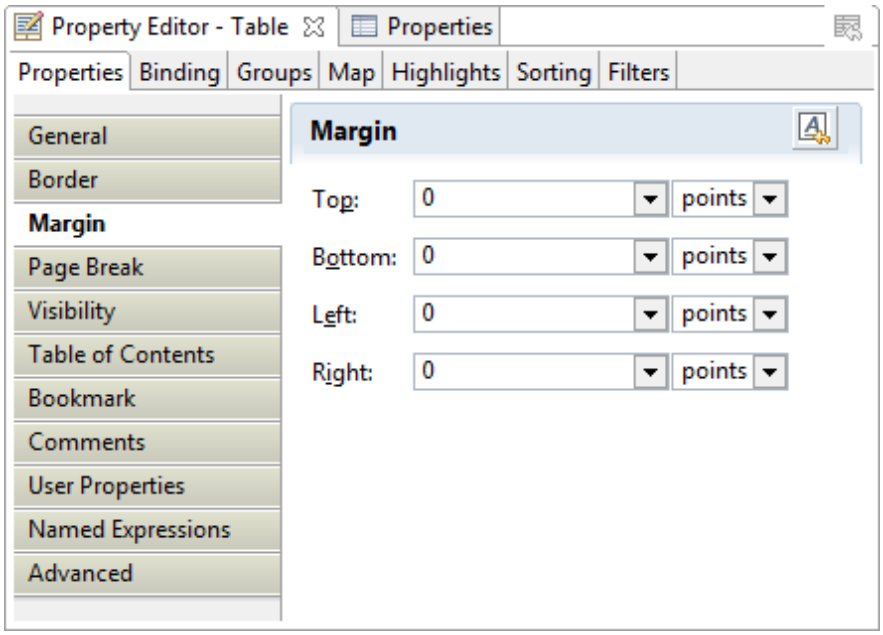


Figura 5

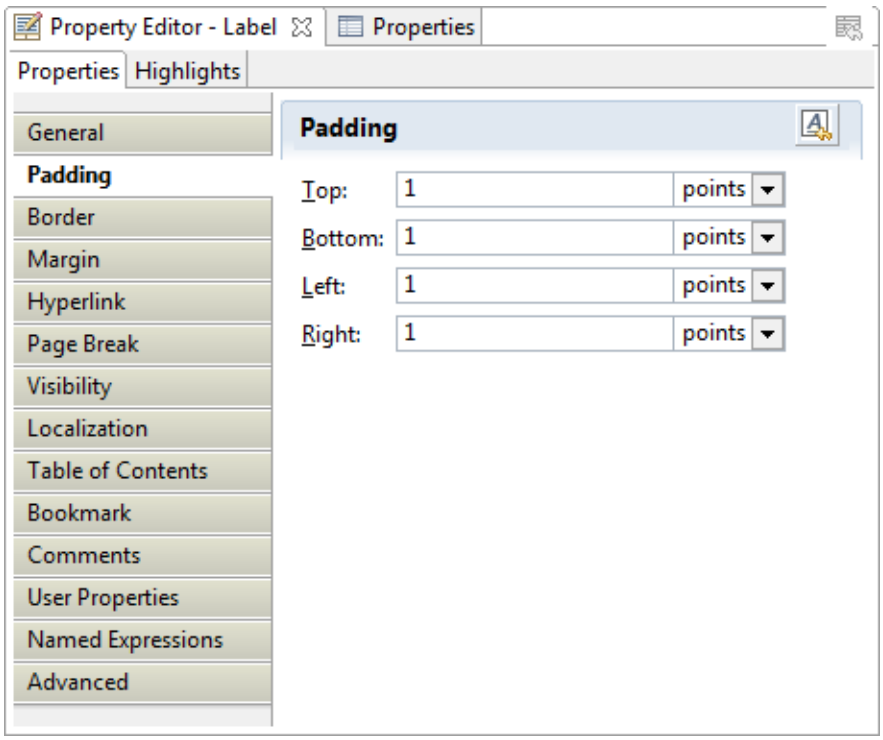


Figura 6

O BIRT fornece três propriedades principais para definir o espaço entre os elementos:

- **Borda (*Border*):** Circunda diretamente o componente. Pode ser visível ou invisível, e sua espessura contribui para o espaçamento geral entre os elementos.
- **Margem (*Margin*):** Corresponde ao espaçamento **externo**, entre a borda do componente e os elementos ao seu redor.
- **Padding (Preenchimento):** Corresponde ao espaçamento **interno**, entre o conteúdo do componente e sua borda.

5.1.4 *Page Break*

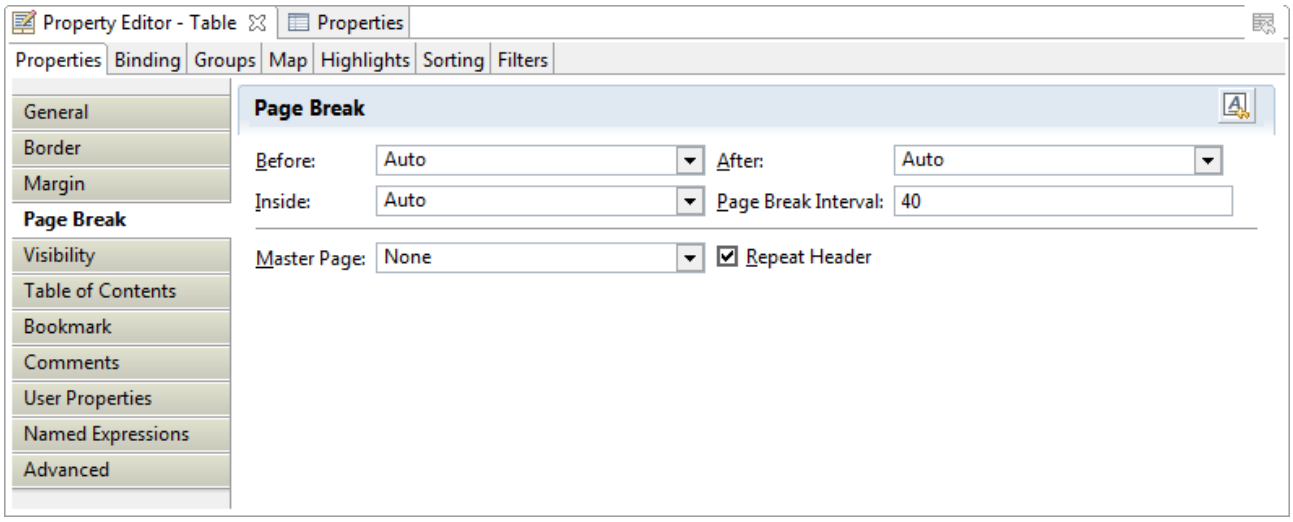


Figura 7

O controle de quebra de página (*Page Break*) possui propriedades que definem onde a quebra pode ocorrer em relação a um componente: *Before*, *After*, *Inside* e *Master Page*.

Para cada uma dessas propriedades, as opções disponíveis são:

- **Auto:** Insere uma quebra de página se for necessário.
- **Always:** Insere uma quebra de página.
- **Avoid:** Evita a quebra de página.

O *Page Break Interval* e *Repeat Header* são características do *table* e *list*. A quebra de página pode ocorrer antes, dentro e após o componente (respectivamente *before*, *inside* e *after*).

- **Master Page:** Campo opcional para especificar uma *Master Page* (modelo de página) diferente da padrão. É desnecessário se o relatório utiliza a mesma configuração em todas as páginas.
- **Repeat Header:** Indica se o cabeçalho da tabela ou lista deve ser repetido em todas as páginas após uma quebra. Por padrão, esta opção vem marcada.
- **Page Break Interval:** Define o número de registros que uma tabela ou lista exibirá antes de forçar uma quebra de página. O valor padrão é **40**.

Atenção:

Por padrão, tabelas e listas quebram a página após **40 registros**, independentemente de haver espaço restante na página para mais itens. Para desabilitar este comportamento e permitir que a página seja preenchida completamente, altere o valor do **Page Break Interval** para **0**.

5.1.5 *Visibility*

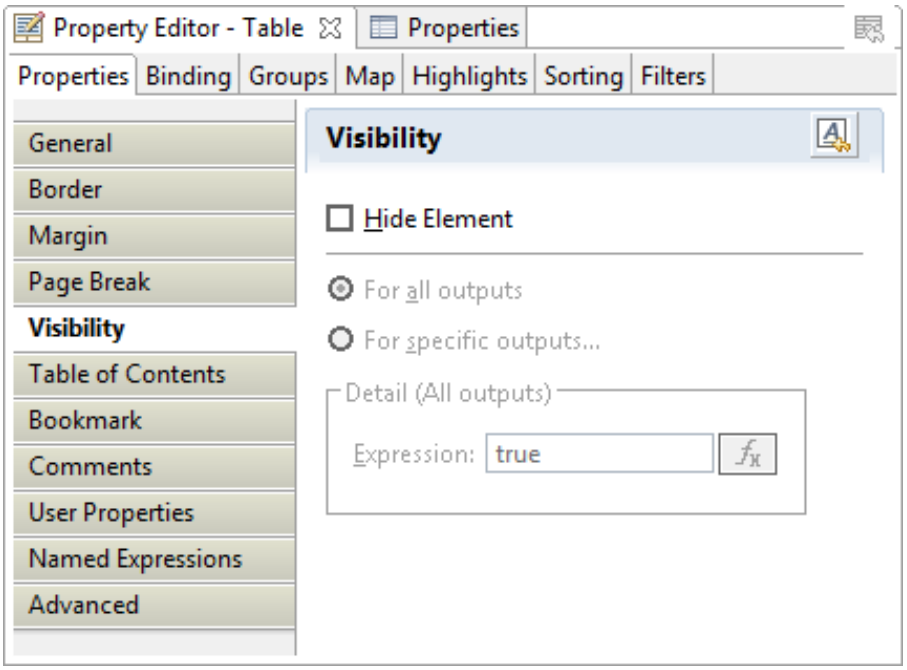


Figura 8

A propriedade de **Visibilidade (*Visibility*)** controla se um componente e seus elementos ficarão ocultos na renderização do relatório.

Hide Element

Esta propriedade permite ocultar um elemento de forma estática. A ocultação pode ser configurada de duas maneiras:

- **For all outputs:** O elemento ficará oculto em todos os formatos de saída (PDF, HTML, etc.), incluindo a pré-visualização (*Preview*).
- **For specific outputs...:** O elemento ficará oculto apenas nos formatos de saída que forem especificados.

Expression

Esta propriedade permite a ocultação **condicional** de um elemento. Deve-se inserir uma expressão que retorne um valor booleano (verdadeiro ou falso). Se o resultado da expressão for **verdadeiro**, o elemento ficará oculto.

Exemplos de Uso

A visibilidade condicional é uma ferramenta flexível e dinâmica, utilizada para:

- Ocultar colunas que se tornam irrelevantes com base nos parâmetros do relatório (ex: ocultar a coluna "Data de Demissão"ao listar apenas funcionários ativos).
- Exibir diferentes seções ou totalizadores com base em um parâmetro de entrada.
- Desativar um elemento temporariamente (usando uma expressão estática como `true`), preservando-o para um possível uso futuro.

Observação:

Elementos ocultos ainda são processados durante a geração do relatório e, portanto, consomem recursos do sistema.

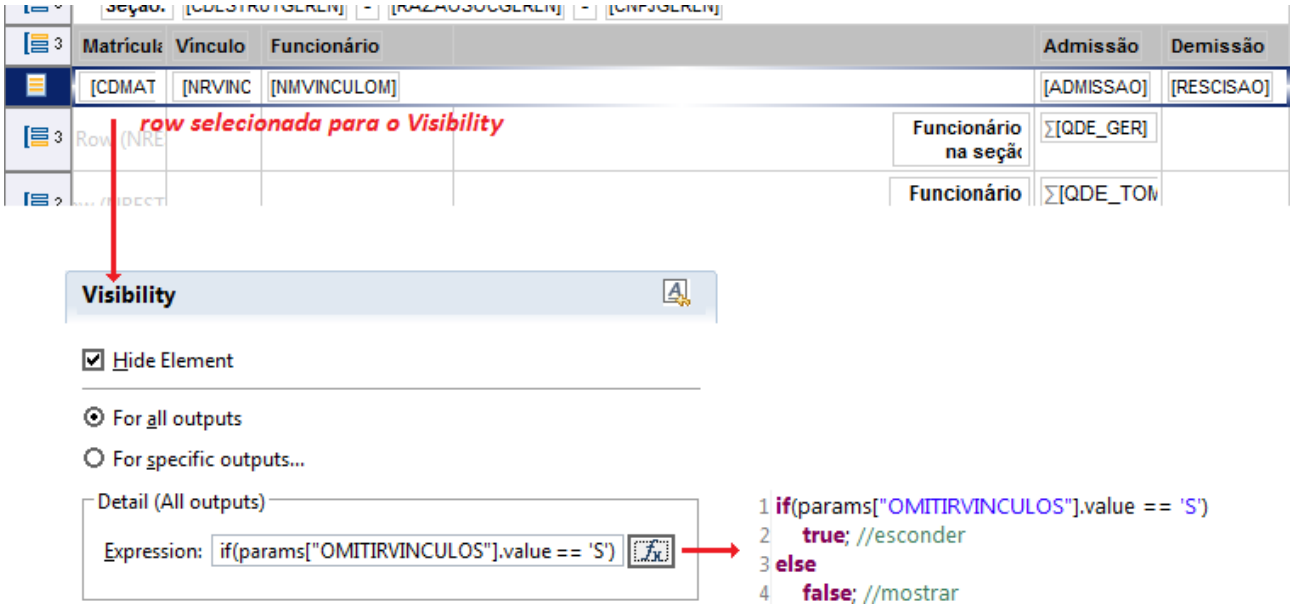


Figura 9

5.1.6 Formato de *Number, Datetime* e *String*

No BIRT, é possível formatar a saída de dados com diversos formatos e máscaras. Esta funcionalidade está disponível exclusivamente para o componente **Data**.

O painel de propriedades (*Properties*) exibe três seções de formatação: **Format Number**, **Format DateTime** e **Format String**. É fundamental utilizar apenas a seção correspondente ao tipo do componente **Data**, mantendo as demais como "Unformatted".

Para verificar rapidamente o tipo de um componente **Data**, dê um duplo clique sobre ele e consulte o campo **"Data Type"**.

Custom

- **Números:** Os caracteres `#` e `0` representam dígitos. O `#` omite zeros não significativos à esquerda, enquanto o `0` os exibe.
- **Strings:** O caractere `@` representa os caracteres do texto.
- **Datas:** Os símbolos de formatação disponíveis podem ser consultados na caixa de exemplo exibida na tela.

- Os códigos de máscara no formato *Custom* diferenciam maiúsculas e minúsculas (*case-sensitive*).

Opção *Locale*

A opção ***Locale*** define a localidade (região/país) para as configurações de formato. Geralmente, esta opção deve ser mantida no valor padrão, **"Auto"**.

Atenção:

- Não existem botões como "OK", "Aplicar"ou "Cancelar". As alterações de formatação são aplicadas **imediatamente** ao componente **Data** selecionado.
- Para remover uma formatação, retorne o tipo de formato (*Format as*) para la opção **"Unformatted"**.

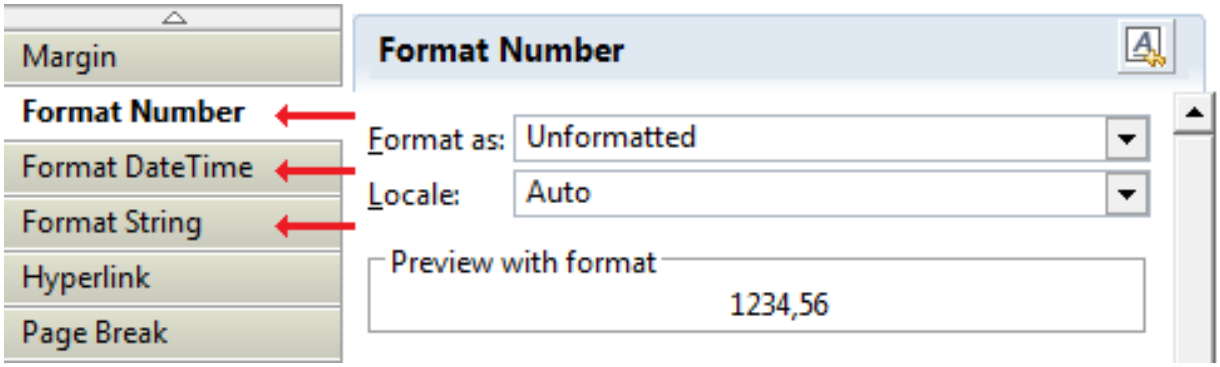


Figura 10

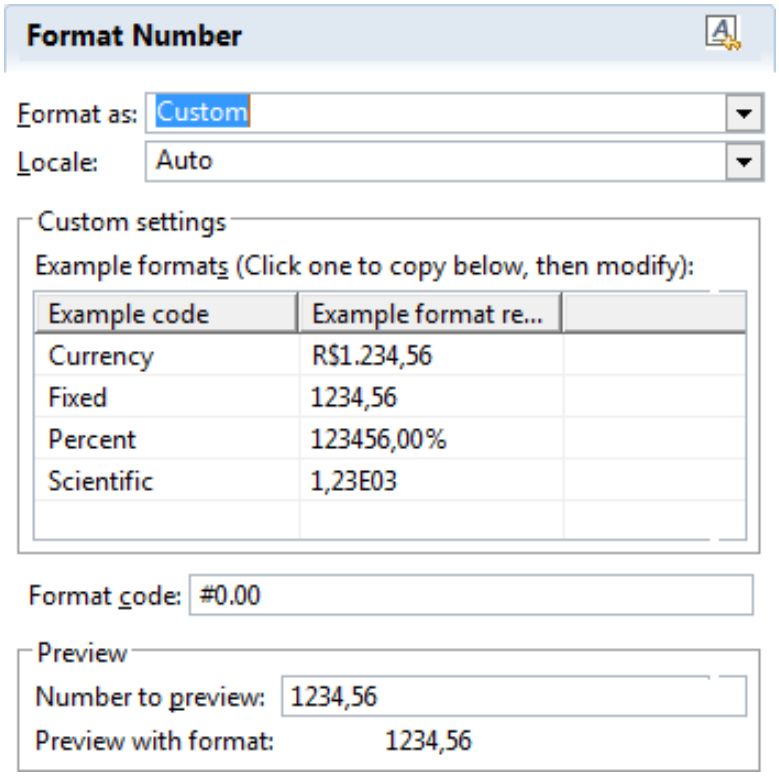


Figura 11

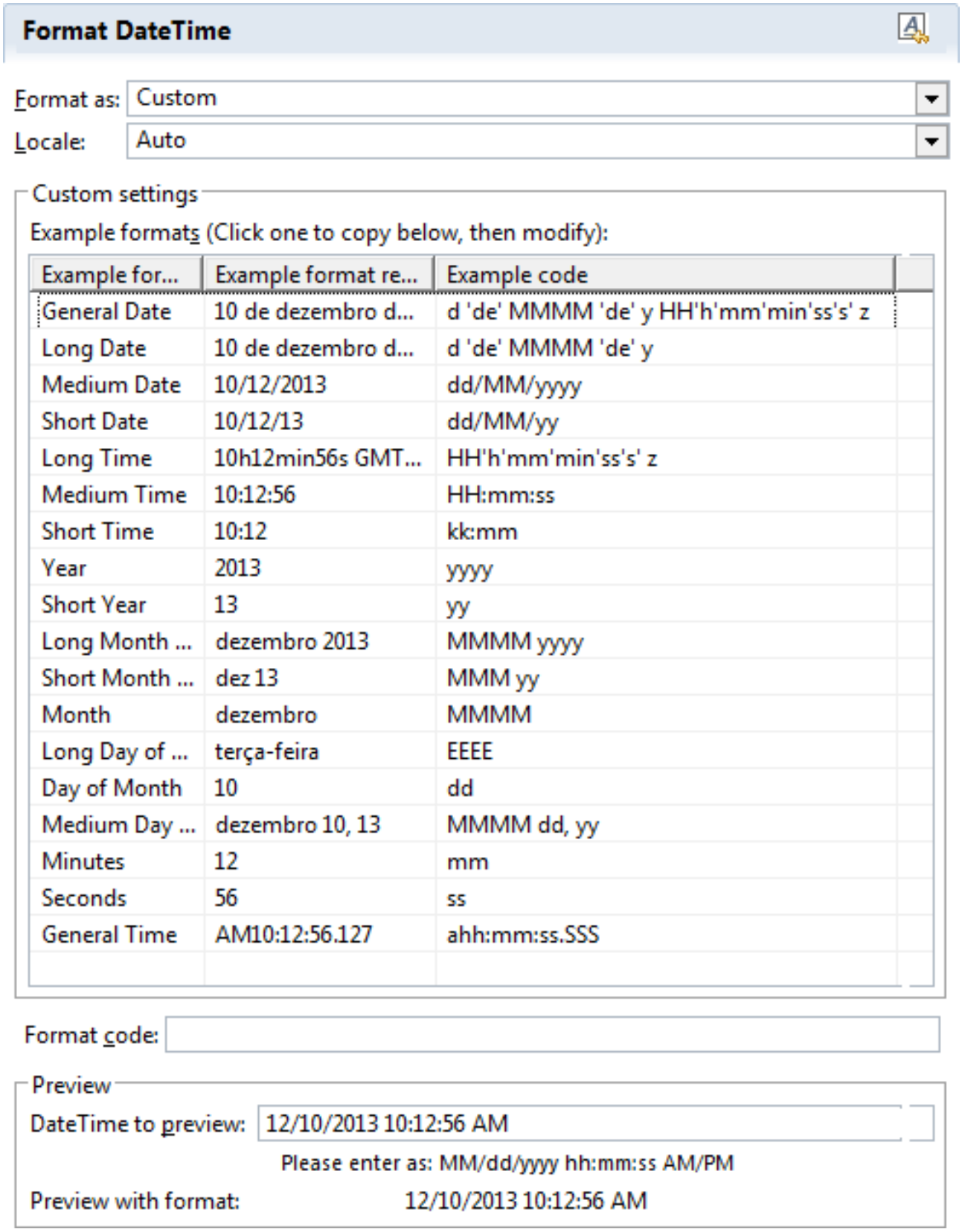

Format DateTime
Format as: Custom
Locale: Auto
Custom settings
Example formats (Click one to copy below, then modify):
Example for... | Example format re... | Example code
General Date | 10 de dezembro d... | d 'de' MMMM 'de' y HH'h'mm'min'ss's' z
Long Date | 10 de dezembro d... | d 'de' MMMM 'de' y
Medium Date | 10/12/2013 | dd/MM/yyyy
Short Date | 10/12/13 | dd/MM/yy
Long Time | 10h12min56s GMT... | HH'h'mm'min'ss's' z
Medium Time | 10:12:56 | HH:mm:ss
Short Time | 10:12 | kk:mm
Year | 2013 | yyyy
Short Year | 13 | yy
Long Month ... | dezembro 2013 | MMMM yyyy
Short Month ... | dez 13 | MMM yy
Month | dezembro | MMMM
Long Day of ... | terça-feira | EEEE
Day of Month | 10 | dd
Medium Day ... | dezembro 10, 13 | MMMM dd, yy
Minutes | 12 | mm
Seconds | 56 | ss
General Time | AM10:12:56.127 | ahh:mm:ss.SSS
Format code:
Preview
DateTime to preview: 12/10/2013 10:12:56 AM
Please enter as: MM/dd/yyyy hh:mm:ss AM/PM
Preview with format: 12/10/2013 10:12:56 AM

Figura 12

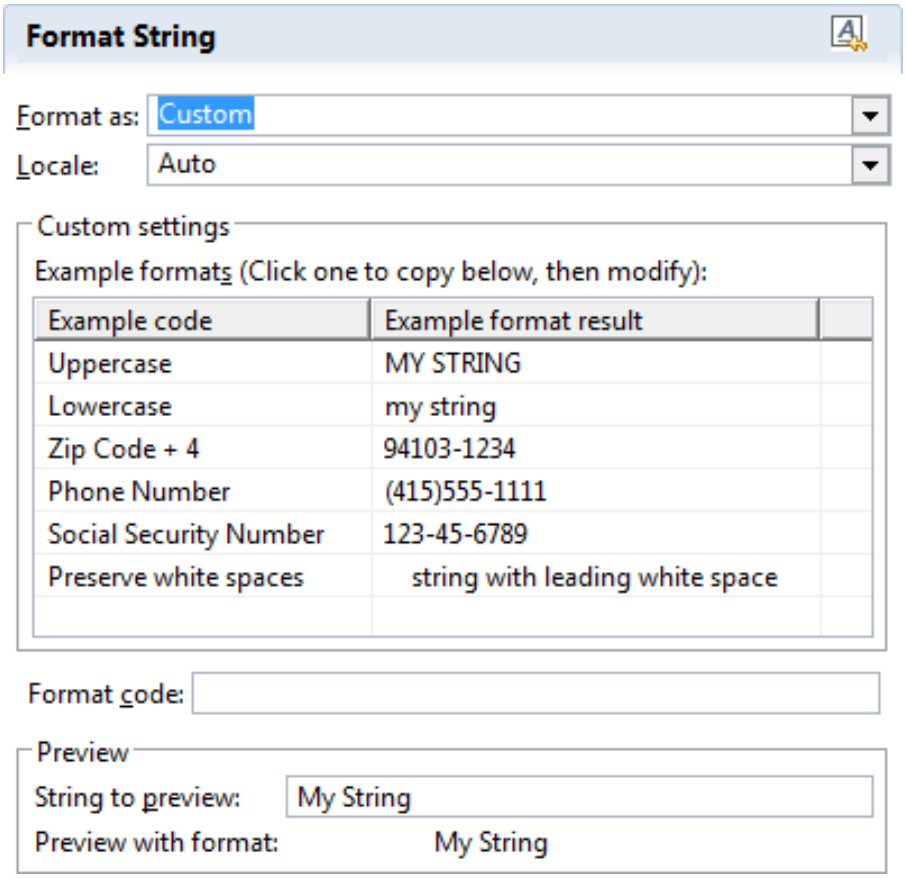


Figura 13

5.2 *HIGHLIGHT LIST*

O *Highlight List* aplica uma formatação específica a um elemento quando a expressão associada a ela retorna o valor **verdadeiro**. Funciona como um mecanismo de formatação condicional.

Seus usos mais comuns incluem:

- Criar um efeito "zebrado"no relatório, alternando a cor de fundo das linhas.
- Alterar a cor da fonte para valores específicos (ex: exibir valores negativos em vermelho).
- Aplicar o estilo negrito a determinados elementos com base em parâmetros ou condições dos dados.

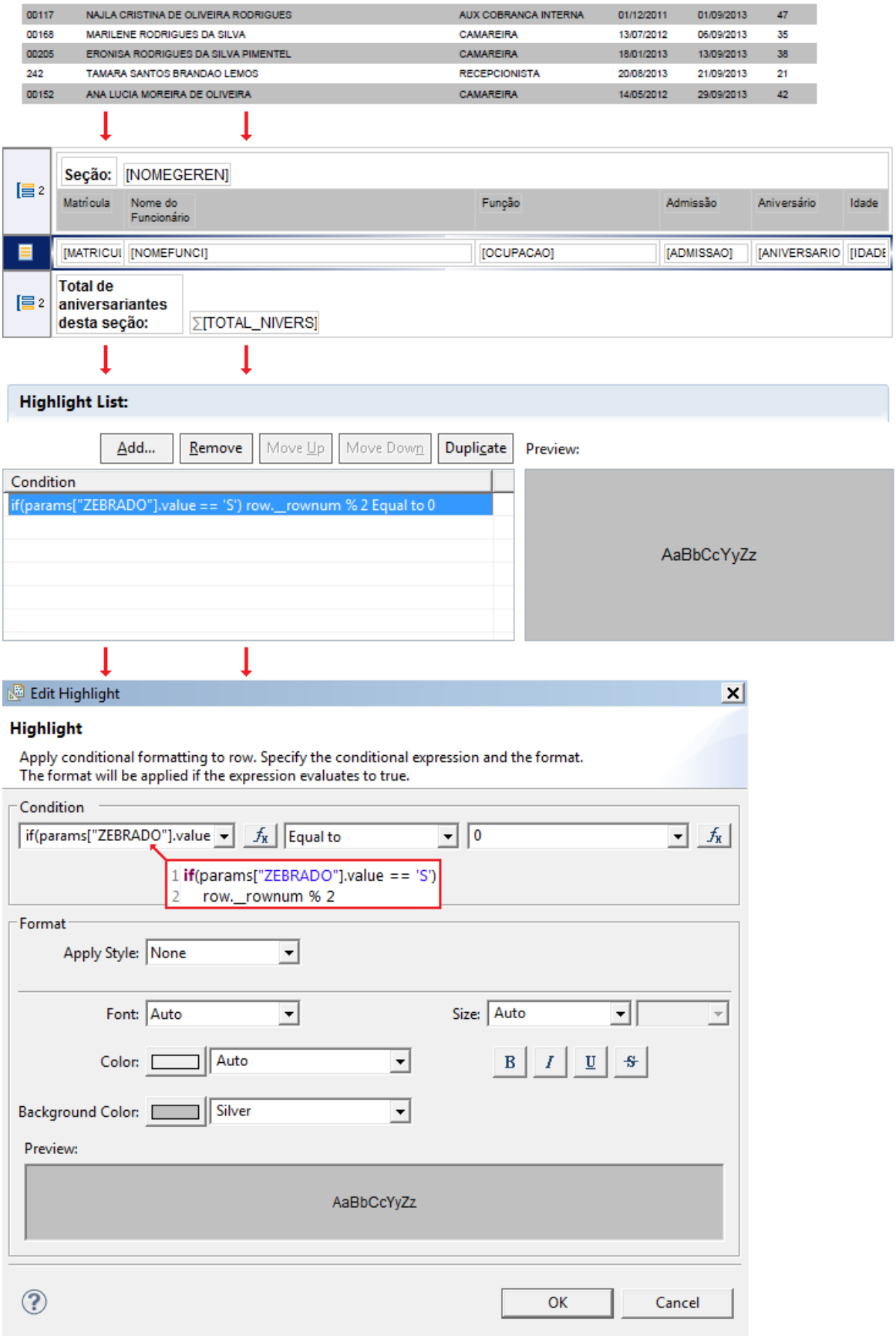

00117 NAJLA CRISTINA DE OLIVEIRA RODRIGUES AUX COBRANCA INTERNA 01/12/2011 01/09/2013 47
00168 MARILENE RODRIGUES DA SILVA CAMAREIRA 13/07/2012 06/09/2013 35
00205 ERONISA RODRIGUES DA SILVA PIMENTEL CAMAREIRA 18/01/2013 13/09/2013 38
242 TAMARA SANTOS BRANDAO LEMOS RECEPCIONISTA 20/08/2013 21/09/2013 21
00152 ANA LUCIA MOREIRA DE OLIVEIRA CAMAREIRA 14/05/2012 29/09/2013 42
Seção: [NOMEGEREN]
Matrícula
Nome do Funcionário
Função
Admissão
Aniversário
Idade
[MATRICUI
[NOMEFUNCI]
[OCUPACAO]
[ADMISSAO]
[ANIVERSARIO
[IDADE
Total de aniversariantes desta seção:
∑[TOTAL_NIVERS]
Highlight List:
Add...
Remove
Move Up
Move Down
Duplicate
Preview:
Condition
if(params["ZEBRADO"].value == 'S') row.__rownum % 2 Equal to 0
AaBbCcYyZz
Edit Highlight
Highlight
Apply conditional formatting to row. Specify the conditional expression and the format.
The format will be applied if the expression evaluates to true.
Condition
if(params["ZEBRADO"].value
Equal to
0
1 if(params["ZEBRADO"].value == 'S')
2 row.__rownum % 2
Format
Apply Style: None
Font: Auto
Size: Auto
Color: Auto
Background Color: Silver
Preview:
AaBbCcYyZz
OK
Cancel

Figura 14

5.3 *MAP*

A propriedade **Map** substitui o valor original de um componente (um campo de *Data*, por exemplo) por um valor estático, predefinido no relatório, quando uma condição é atendida.

Seu funcionamento é similar ao da propriedade **Destaque (*Highlight*)**. A diferença é que, em vez de aplicar uma formatação, o *Map* substitui o valor do componente. A estrutura de criação de regras (com base em uma expressão) é a mesma, mas o resultado é a substituição do valor, e não a aplicação de um estilo.

Assim como no *Highlight*, a **condição** para a substituição pode ser dinâmica (baseada em expressões). No entanto, o **novo valor** a ser exibido é sempre estático, ou seja, é o valor que foi definido na própria regra de mapeamento.

Expression Builder

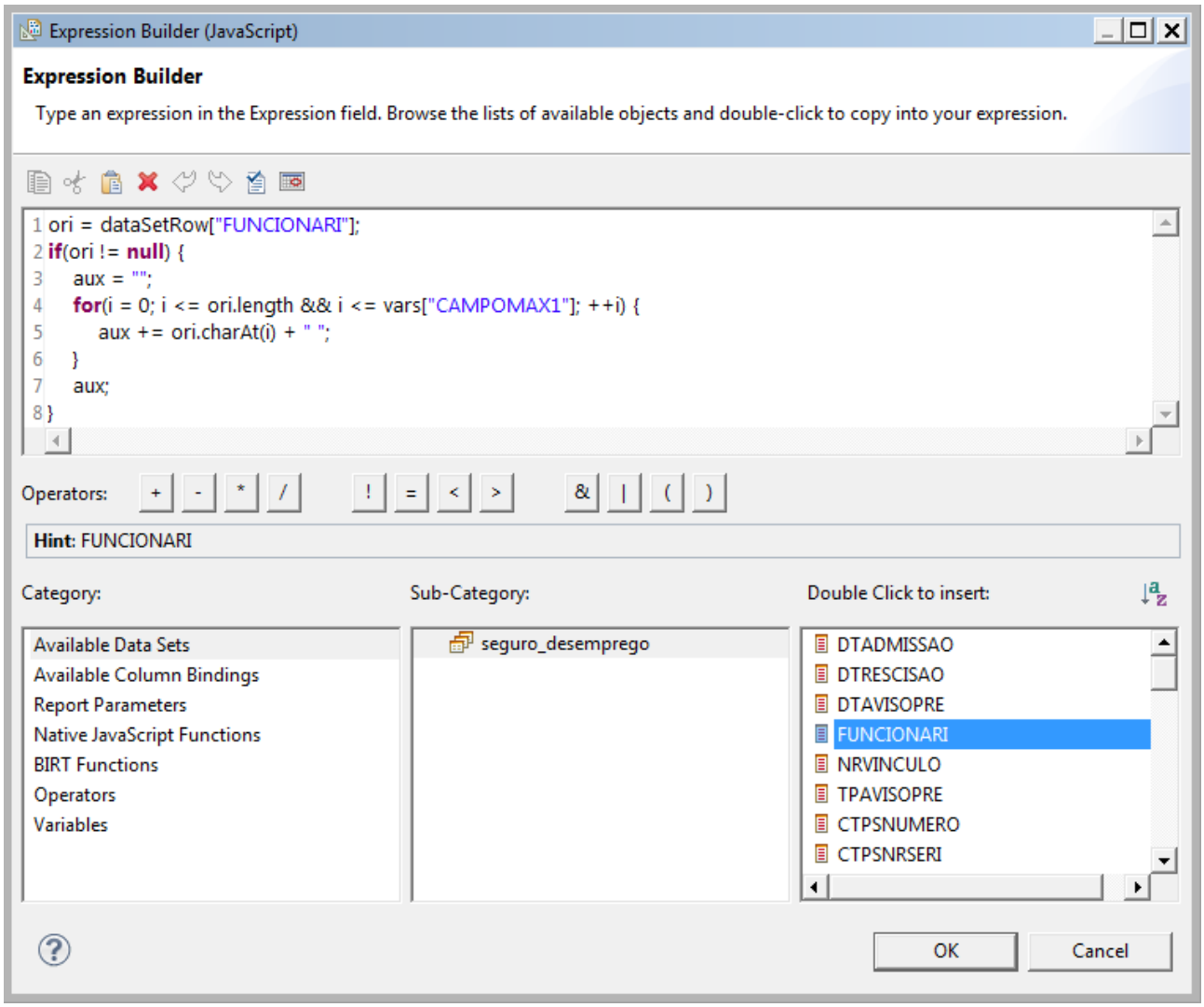


Figura 15

O ***Expression Builder*** (Construtor de Expressões) é a interface utilizada para construir lógicas e rotinas com base na sintaxe JavaScript, oferecendo grande flexibilidade em diversas áreas do BIRT. Ele também fornece um explorador de objetos que auxilia o usuário a localizar funções e constantes.

Pontos importantes sobre o uso do *Expression Builder*:

- **Sintaxe JavaScript:** A sintaxe é baseada em JavaScript. Embora possam existir pequenas diferenças em alguns métodos específicos, a maior parte do uso comum é idêntica. Muitos exemplos e boas práticas podem ser encontrados em fóruns da comunidade.
- **Delimitador de Escopo (...):** Blocos com apenas uma instrução não exigem os delimitadores, mas blocos com duas ou mais instruções os requerem. Como boa prática para clareza e depuração do código, recomenda-se utilizar os delimitadores de escopo em todos os blocos.

- **Comando *return*:** O uso do comando return não é necessário. O valor da última variável, função ou método no bloco de instruções é retornado implicitamente.
- **Ponto e vírgula (;):** O uso do ponto e vírgula (;) é facultativo na última instrução do bloco.

```
if( a > b ) //uso do {} facultativo
    --a;    //uma única instrução
else
    ++b;    //uma única instrução
//ou
for( i = 0; i < 10; ++i )   //uso do {} facultativo
    ++b;                    //uma única instrução
```

```
if( a > b ) {   //uso do {} obrigatório
    --a;        //uma única instrução
} else {
    a = b;      //várias instruções
    ++b;
}
```

```
if( a > b )     //NÃO FUNCIONA! uso do {} obrigatório
    --a;        //uma única instrução
else {
    a = b;      //várias instruções
    ++b;
}
```

```
if( a > b )
    a;
else
    b;

/* return b; */    //Não usar return
```

```
b = -5;
for( i = 0; i < 10; ++i ) {
    ++b;
}
b  //na ÚLTIMA instrução o ; é facultativo
/* return b; */    //Não usar return
```

Figura 16

Master Page

A **Página Mestra (*Master Page*)** é um modelo que define as propriedades básicas de uma página, como orientação e margens. O BIRT utiliza por padrão a "Simple MasterPage", que é indicada para relatórios simples ou direcionados à *web*. Geralmente, basta configurar a página mestra padrão, não sendo necessário criar novas.

É possível associar componentes específicos a diferentes *Master Pages*, permitindo, por exemplo, que uma parte do relatório tenha orientação retrato e outra, paisagem.

Para criar uma nova *Master Page*:

1. Na guia **Outline**, clique com o botão direito sobre o item **Master Pages** e selecione "Insert element".
2. Um novo item será criado (ex: "NewSimpleMasterPage").
3. Com a aba **Master Page** da área de trabalho ativa, selecione o novo item criado para que suas propriedades sejam carregadas nos painéis e possam ser editadas.

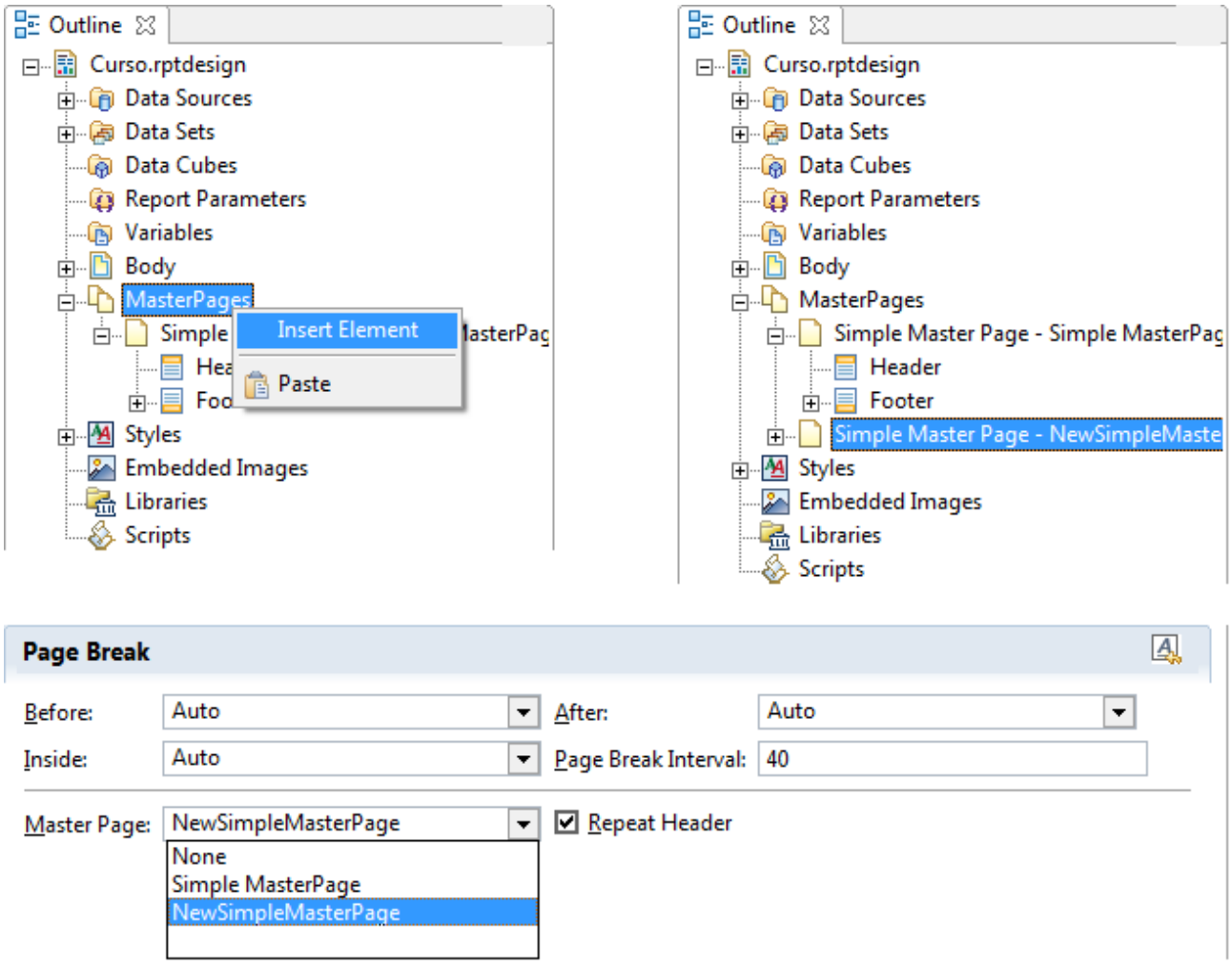


Figura 17

Para associar uma *Master Page* a um componente (como uma *Table*), selecione-o, acesse a propriedade **Page Break** e escolha a *Master Page* desejada no campo "Master Page".

Os painéis de configuração da página são, em geral, simples e intuitivos. A seguir serão brevemente detalhadas suas propriedades:

7.1 GENERAL

Neste painel são definidas as configurações de tamanho da página, orientação (retrato/paisagem), cor de fundo e as alturas do cabeçalho e do rodapé.

7.2 BORDER

Permite a configuração das bordas da página. É possível definir cor, espessura e estilo para as bordas superior, inferior, direita e esquerda de forma independente. Uma pré-visualização do resultado é exibida no painel.

7.3 MARGIN

Define as margens superior, inferior, direita e esquerda da página. A altura especificada para o cabeçalho e o rodapé é aplicada a partir do limite interno destas margens.

7.4 HEADER/FOOTER

Controla a exibição do cabeçalho e do rodapé na primeira e na última página do relatório.

7.5 COMMENTS

Campo opcional destinado à inserção de comentários ou anotações do usuário.

Primeiros Passos

Esta seção demonstra como realizar as configurações básicas para se fazer um relatório no BIRT.

Para demonstração será utilizado o exemplo baseado na tabela *gpe_vinculom* do *teknisa_saas*.

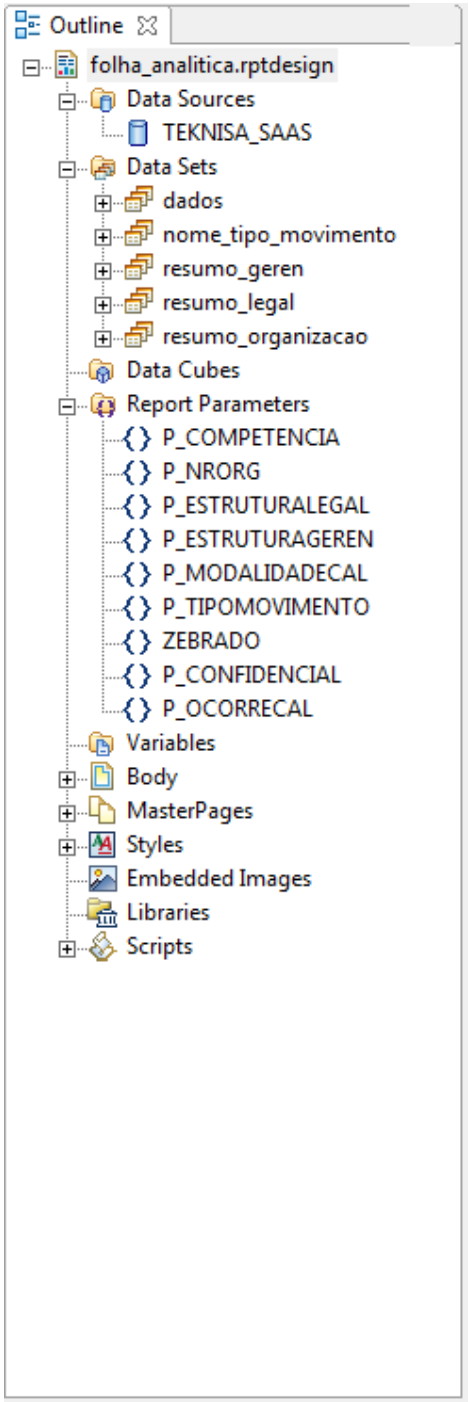


Figura 18

8.1 CONFIGURANDO O *DATA SOURCE*

1. Na aba **Data Explorer** clique com o botão direito em *Data Sources* e escolha *New Data Source*.

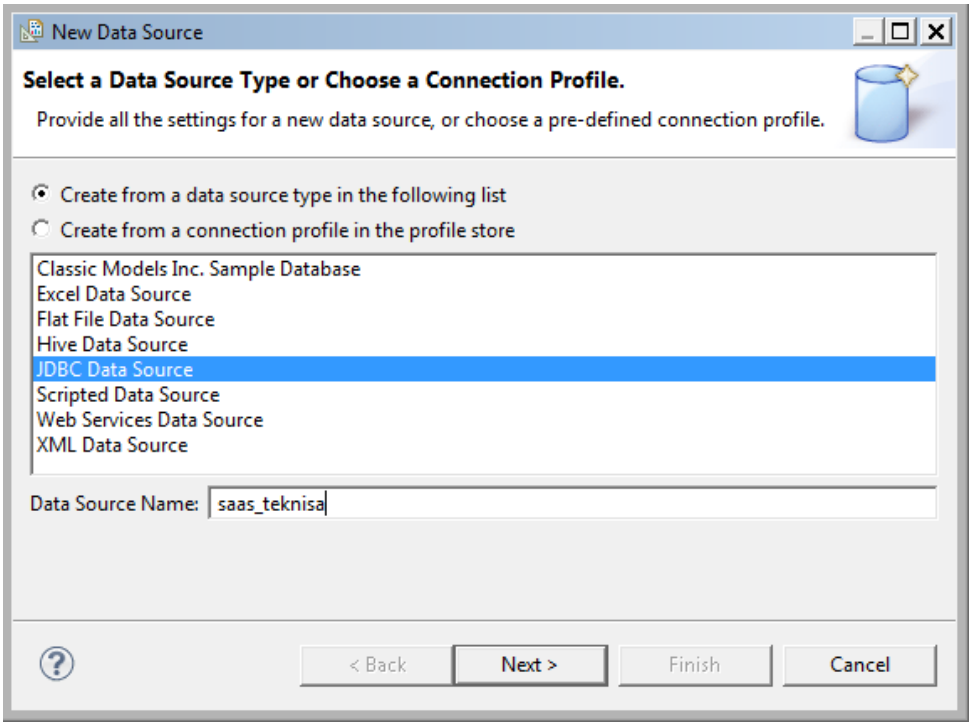


Figura 19

2. Na janela **New Data Source** selecione *"Create from a data source in the following list"*, *"JDBC Data Source"* e um nome para o *Data Source* em *"Data Source Name"*.
3. Clique no botão *next*.
4. Na nova janela aberta, digite os parâmetros para a conexão no banco de dados.

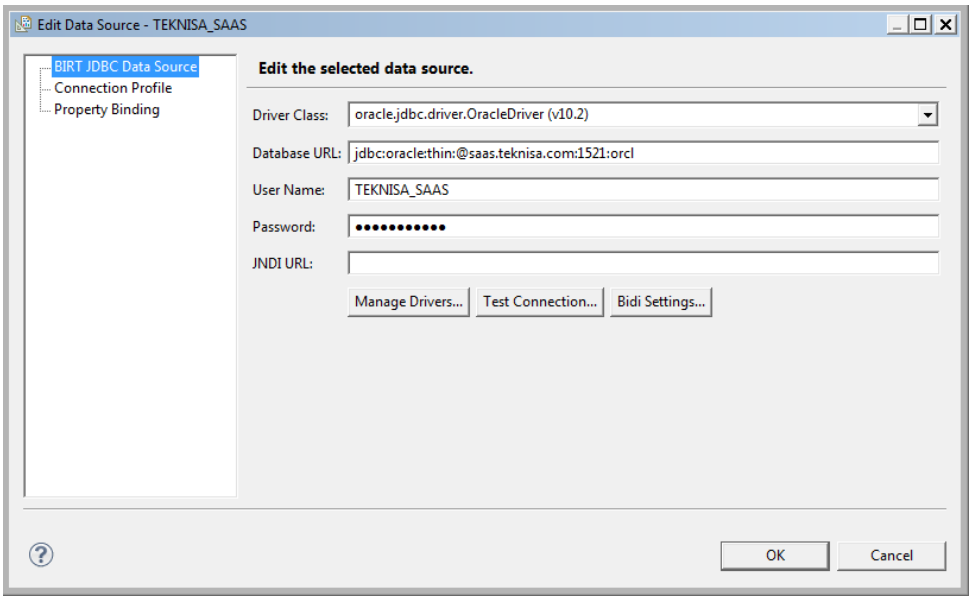


Figura 20

5. Clique no botão **"Test Connection..."** para testar se a conexão está corretamente configurada. Se a mensagem que aparecer informar sucesso clique em **"OK"**. Caso contrário reveja as configurações.

8.2 CONFIGURANDO O *DATA SET*

1. Na aba **Data Explorer** clique com o botão direito em *Data Sets* e escolha *New Data Set*;

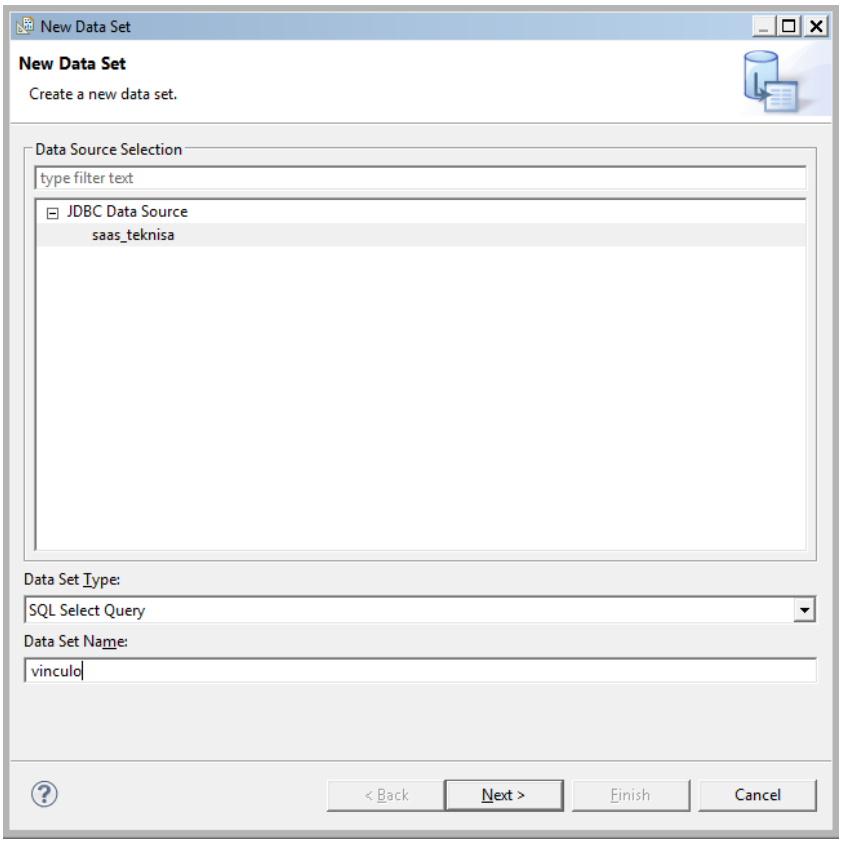


Figura 21

2. Em *"Data Set Type"* selecione *"SQL Select Query"*;
3. Em *"Data Set Name"* digite o nome para o *data set*;
4. Clique no botão *"Next"*;
5. Na nova janela coloque a *query* na área *"Query Text"* (segue abaixo o texto da *query* utilizada neste exemplo);

```
1
2  SELECT
3    VM.NRVINCULOM NRO,
4    VM.NMVINCULOM NOME,
5    VM.DTADMISSAOVINC ADMISSAO,
6    VM.DTRESCISAOVINC RESCISAO/*,
7    VM.**/
8  FROM GPE_VINCULOM VM
9  WHERE VM.NRORG = 1002
10   AND VM.DTADMISSAOVINC >= '01/07/2013'
11   --AND VM.DTRESCISAOVINC IS NULL
12 ; --O BIRT nao aceita este caracter
```

```
13
14
```

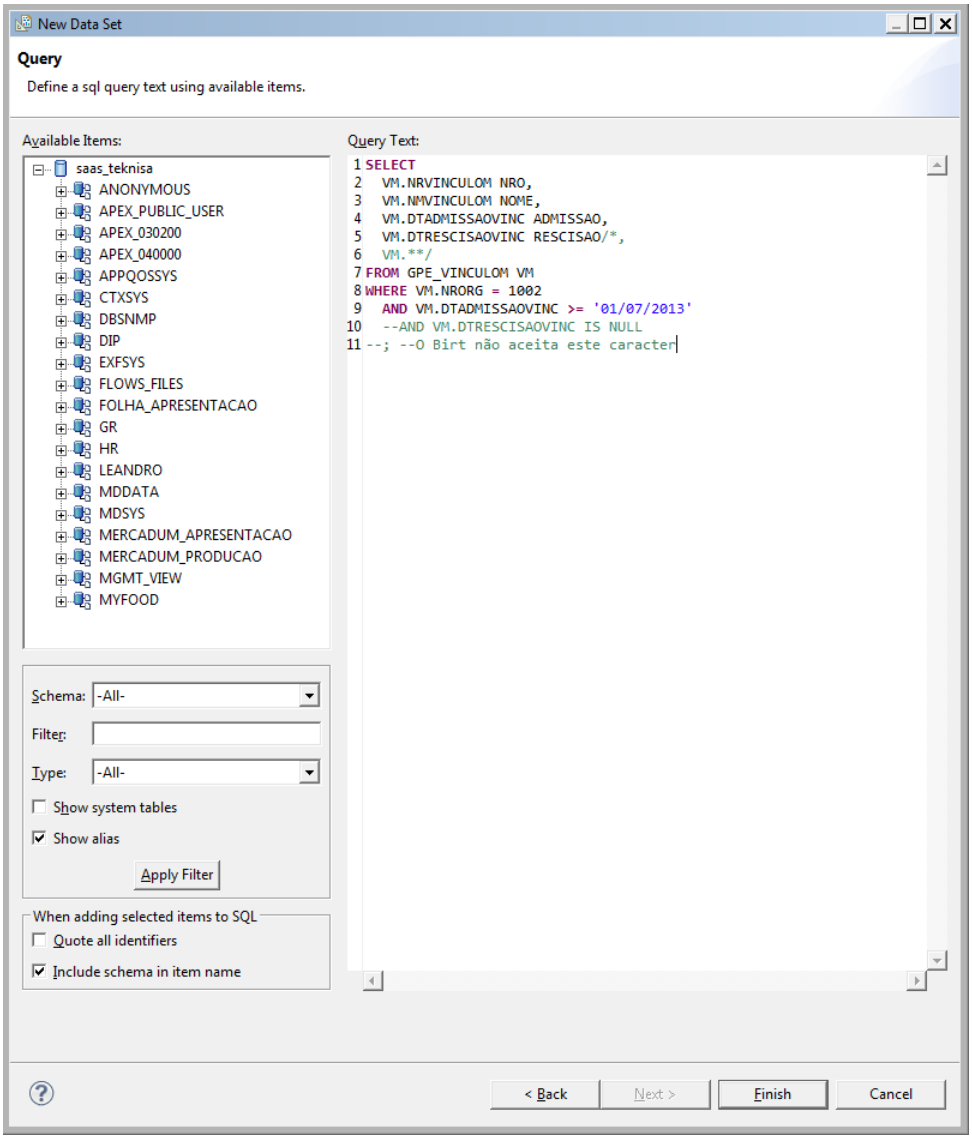


Figura 22

6. Não interfira nas demais opções da tela, clique em *"Finish"*;

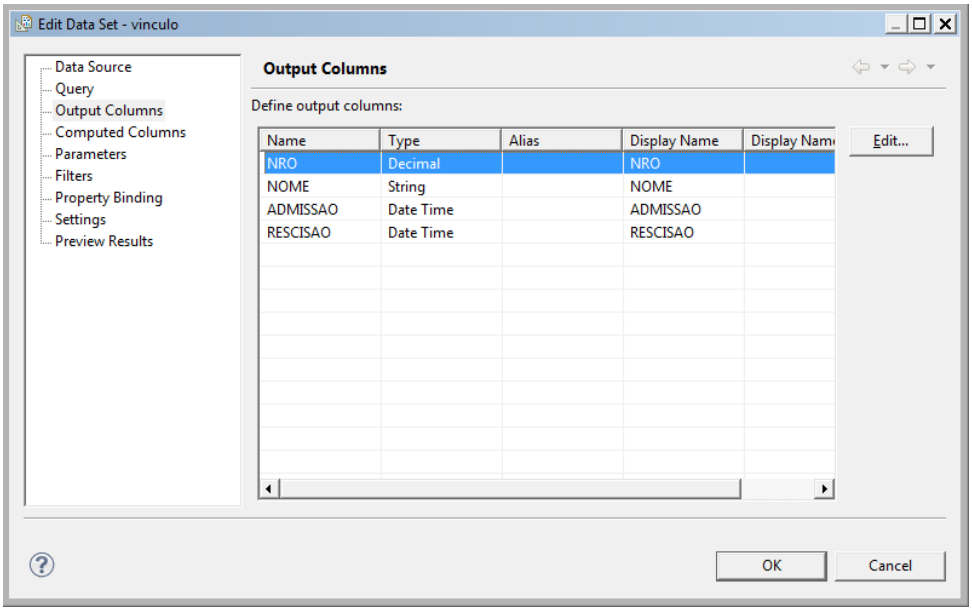


Figura 23

A nova janela aberta fornece opções de configuração do seu *data set*, tais como parâmetros, colunas computadas, filtros, etc.

7. Se desejar verificar se está tudo funcionando selecione *"Preview Results"* na árvore do lado esquerdo;

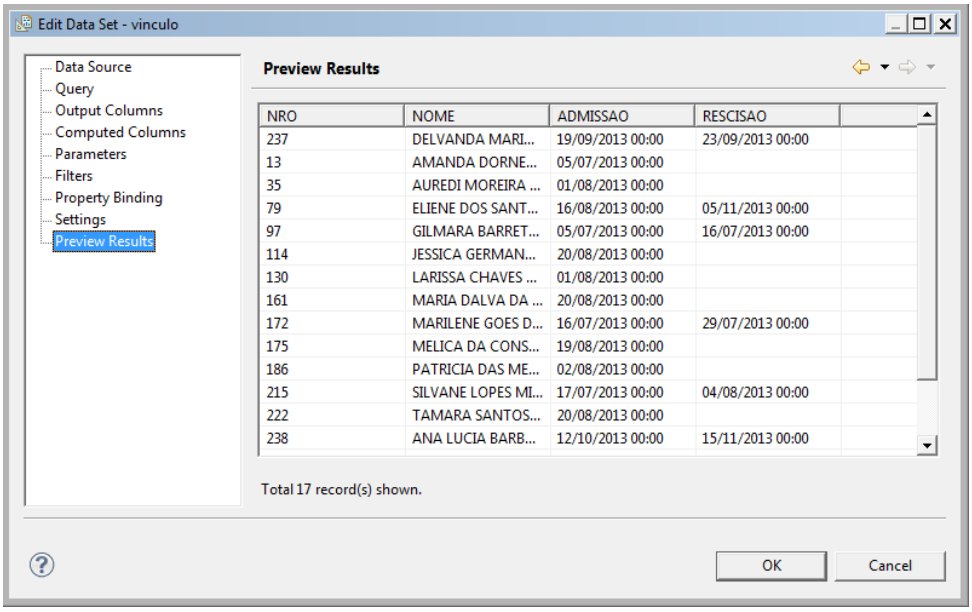


Figura 24

8. Clique em **"OK"**;

8.3 INSERINDO E CONFIGURANDO UMA TABELA NO RELATÓRIO

Esta parte explica como inserir uma tabela que retornará os dados do *data set* recém criado.

1. Na parte superior esquerda da interface do BIRT selecione a aba **"Data Explorer"**;
2. No item da árvore **"Data Sets"** clique e arraste o *data set* criado (neste exemplo ele se chama **"vinculo"**) nos passos anteriores para a área do relatório;

O BIRT automaticamente criará uma **Table** associada ao *data set* em questão com todos os campos gerados pelo *data set* e com seus respectivos rótulos. Isto é muito útil para pequenos relatórios ou para se iniciar um relatório mais complexo.

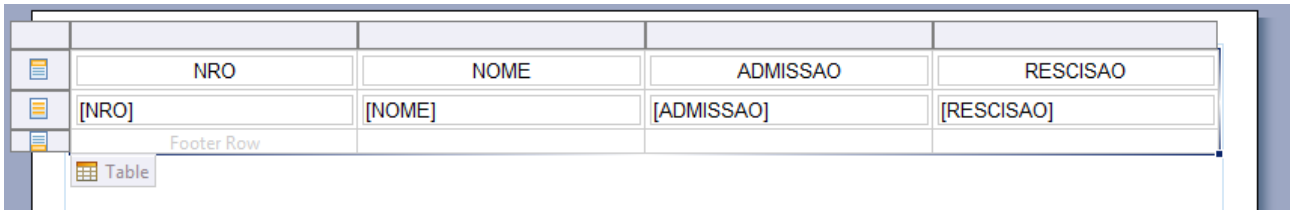


Figura 25

Ao se selecionar o elemento **Table** criado acima percebe-se os rótulos à esquerda indicando *"Header"*, *"Detail"* e *"Footer"*. Mais adiante será explanado detalhadamente estes elementos.

8.4 RENDERIZANDO UM RELATÓRIO

Há dois modos básicos de se visualizar um relatório: pela aba **"Preview"** da área de trabalho do BIRT, e o outro método é gerar um arquivo (*renderizar*) *pdf*, *doc*, *xls*, *html*, etc. O mais comum é *renderizar* o relatório em *pdf* como arquivo final, pronto para impressão.

A vantagem da visualização via *Preview* é a possibilidade de ver os retornos de erros que eventualmente podem acontecer em um ou mais componentes do relatório. Os erros virão no final da pré-visualização.

Para visualizar via *Preview*, siga os passos:

1. Clique na aba **"Preview"** da área de trabalho.

Para *renderizar* em *pdf*, siga os passos:

1. Clique no menu **"Run"**, em seguida **"View Report"** e **"7 As PDF"**.

Em uma nova janela será exibido o relatório *renderizado*. Pode-se salvá-lo em **PDF** e imprimi-lo diretamente desta janela. Eventuais erros serão suprimidos nesta visualização.

Script

Além do *Expression Builder*, o BIRT permite utilizar *scripts* em *JavaScript* em quase todos os seus componentes com eventos tais como *beforeopen*, *afterclose*, entre outros.

Na ajuda do BIRT ou no site do Eclipse é encontrado todo o repertório de comando e suas respectivas sintaxes.

Uma das muitas tarefas que é possível realiza com os *scripts* é a substituição dinâmica de uma palavra (ou expressão) na *query* por outra *string* gerada dinamicamente ou recuperada de um parâmetro ou ambos.

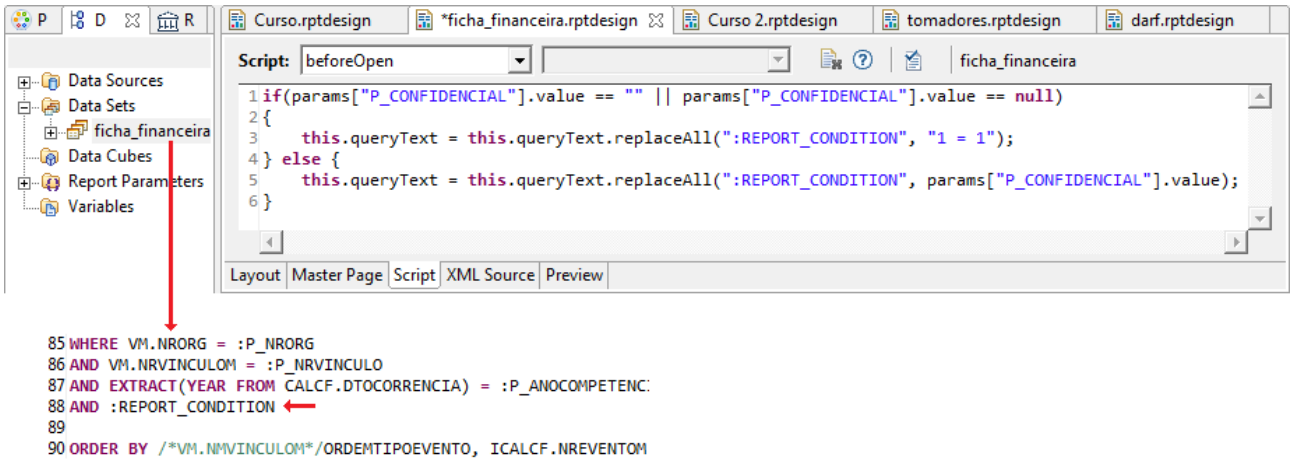


Figura 26

Note que o *script* de exemplo acima está aplicado ao *dataset*. Com o *dataset* e a aba **Script** selecionados, o evento **beforeOpen** foi escolhido para conter o código.

Este código substitui a expressão **:REPORT_CONDITION** por um parâmetro passado ao relatório. Adicionalmente, ele realiza uma verificação: se o parâmetro não possuir valor, a substituição é feita pela expressão **1 = 1**, garantindo que a condição da consulta permaneça válida.

Chart

O **BIRT** permite a inserção de gráficos em seus relatórios, muito similar ao Excel.

As imagens de exemplo a seguir foram extraídas da documentação oficial do Eclipse.

O *chart* precisa de um *data source* e um *data set*.

Para criar um *chart*, siga o passo a passo abaixo:

1. Usando o *Pallete*, insira um *Chart* no relatório. Na janela aberta, é possivel o tipo e subtipo do gráfico, conforme ilustração a seguir.
2. Clique em **next** para prosseguir

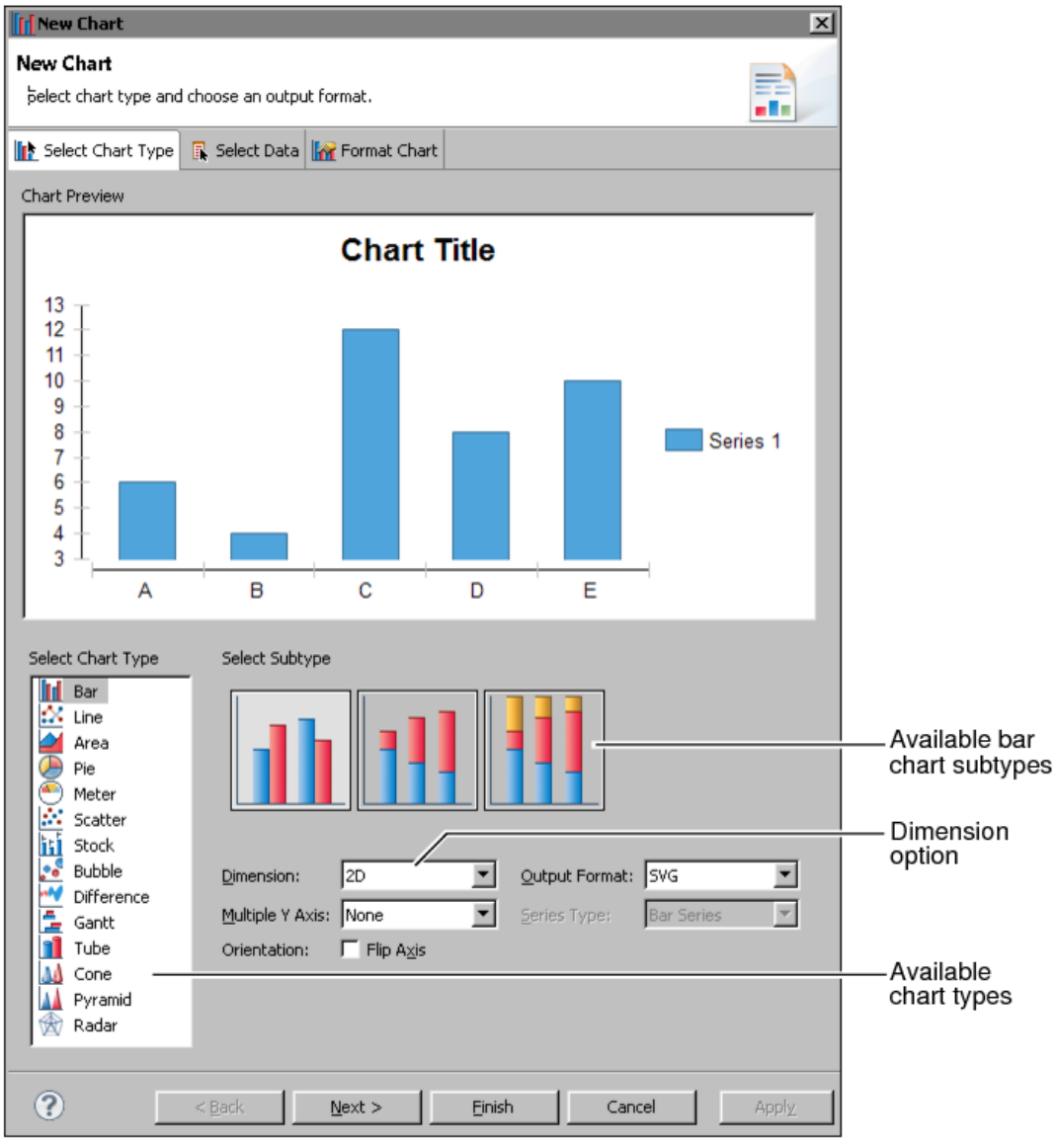


Figura 27

3. Selecione o *Data Set* em uso no **Select Data** → **Use data from**.

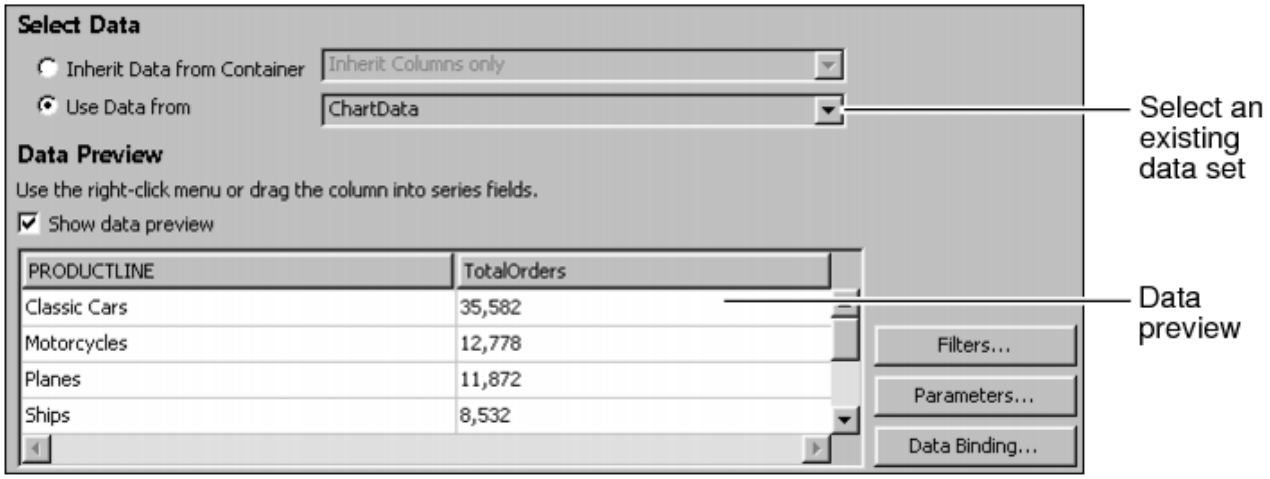


Figura 28

4. O campo destinado aos valores que serão os rótulos do eixo X deve ser colocado no **Category (X) Series:**. Para isso, é possível utilizar o *Builder Expression* ou clicar e arrastar o rótulo do campo da tabela abaixo, conforme ilustrado a seguir.

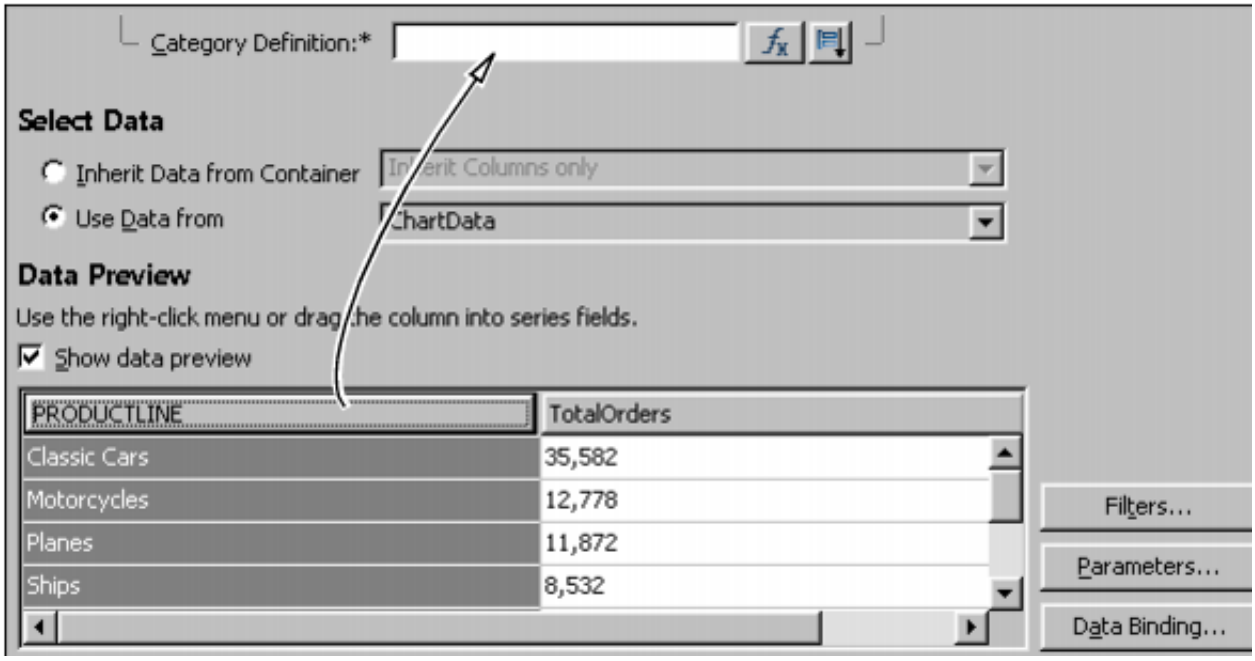


Figura 29

5. Os valores mensurados no eixo Y são extraídos do campo associado ao rótulo. Provavelmente um campo totalizador (*sum*). Insira este campo no **Value (Y) Series:**, semelhante ao eixo X.
6. Clique em **"next"** para prosseguir.

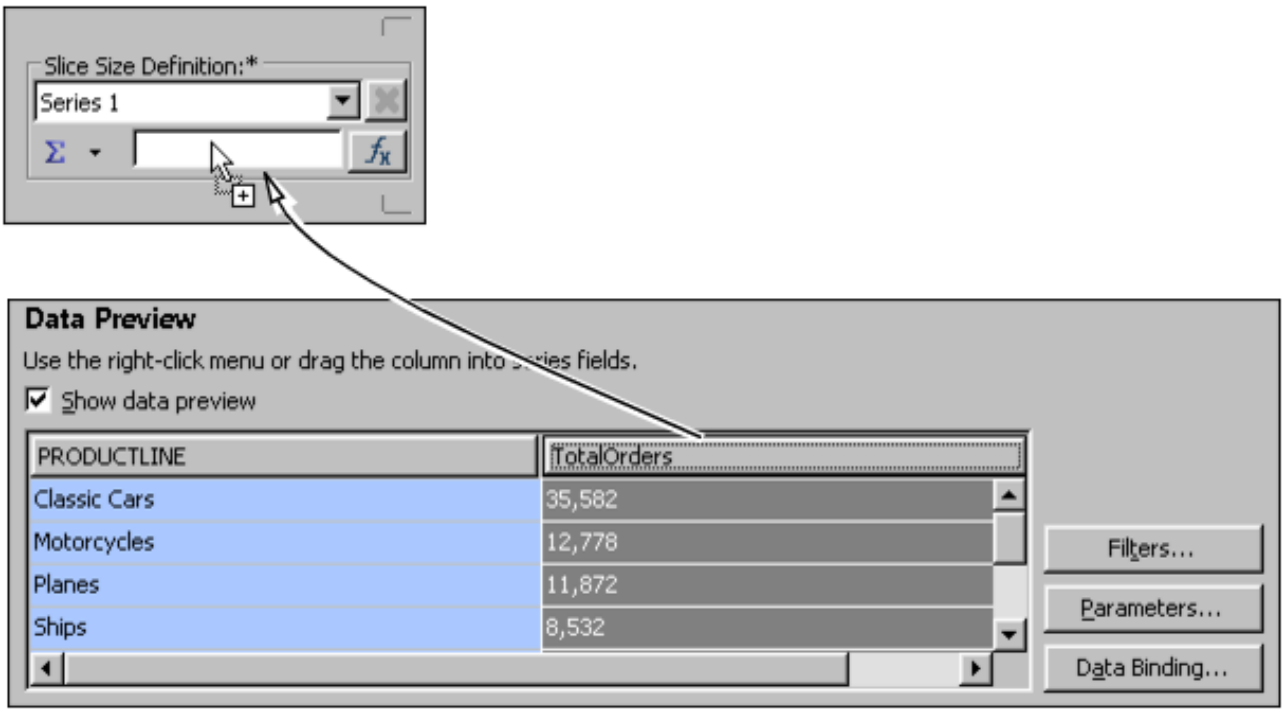


Figura 30

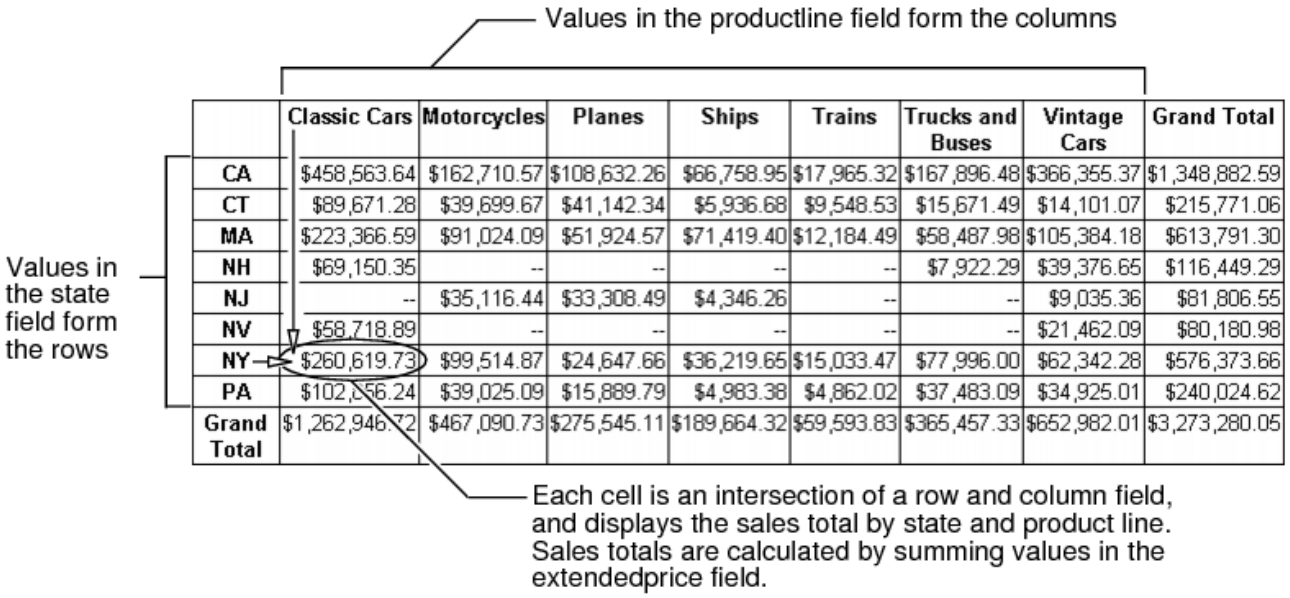
7. Na janela apresentada, é visualizado o acabamento final do *chart*.
8. Clique em **finish**.

Cross Tab

A *Cross Tab* exibe dados em uma matriz semelhante a uma folha de cálculo, como no Excel. Como uma planilha, a célula é ideal para resumir dados em um formato compacto e conciso.

Existe um conjunto de dados listados no lado esquerdo da matriz e outro conjunto de dados listados na parte superior da matriz, no cruzamento de linha e coluna estão relacionados os agrupamentos. Estes agrupamentos exibem resumo, agregação, valores como somas, contagens ou médias.

A seguir uma ilustração de um *Cross Tab* (extraída do site oficial do Eclipse):



	Classic Cars	Motorcycles	Planes	Ships	Trains	Trucks and Buses	Vintage Cars	Grand Total
CA	$458,563.64	$162,710.57	$108,632.26	$66,758.95	$17,965.32	$167,896.48	$366,355.37	$1,348,882.59
CT	$89,671.28	$39,699.67	$41,142.34	$5,936.68	$9,548.53	$15,671.49	$14,101.07	$215,771.06
MA	$223,366.59	$91,024.09	$51,924.57	$71,419.40	$12,184.49	$58,487.98	$105,384.18	$613,791.30
NH	$69,150.35	--	--	--	--	$7,922.29	$39,376.65	$116,449.29
NJ	--	$35,116.44	$33,308.49	$4,346.26	--	--	$9,035.36	$81,806.55
NV	$58,718.89	--	--	--	--	--	$21,462.09	$80,180.98
NY	$260,619.73	$99,514.87	$24,647.66	$36,219.65	$15,033.47	$77,996.00	$62,342.28	$576,373.66
PA	$102,256.24	$39,025.09	$15,889.79	$4,983.38	$4,862.02	$37,483.09	$34,925.01	$240,024.62
Grand Total	$1,262,946.72	$467,090.73	$275,545.11	$189,664.32	$59,593.83	$365,457.33	$652,982.01	$3,273,280.05

Figura 31

Um *Cross Tab* precisa de um *Data Cube* para funcionar, e por sua vez, o *Data Cube* precisa de um *Data Set*.

O *Data Cube* é quem fornece os dados ao *Cross Tab*. Nele, é definido quais os resultados de uma coluna serão a linhas ou colunas do *Cross Tab*.

No exemplo acima, os valores "CA", "CT", "MA", etc., da coluna **STATE** formam a coluna esquerda do *Cross Tab*, enquanto os valores "Classic Car", "Motorcycles", "Planes", etc., da coluna **PRODUCTLINE** formam as colunas da tabela.

Veja o *Preview Results* do *Data Set* no exemplo da figura 32.

Na interseção da linha (formada por valores de **STATE**) e coluna (formada por valores de **PRODUCTLINE**) está a somatória dos valores de **EXTENDED_PRICE** (esta coluna é uma *Computed Column*. Ou seja, não existe no banco de dados).

Sendo assim, a *Cross Tab* exibirá nas interseções uma agregação (somatória, média, contagens, etc.) dos valores correspondentes ao registro que contém simultaneamente os valores da linha superior e da coluna esquerda da *Cross Tab*.

Não é possível exibir os valores na *Cross Tab* sem uma agregação qualquer, uma vez que é possível existir mais de uma ocorrência da interseção da linha superior e coluna esquerda da *Cross Tab*. Por exemplo, é

possível que exista mais de um registro contendo simultaneamente um **PRODUCTLINE** “Planes” e **STATE** “CA”. Consequentemente, haverá um **EXTENDED_PRICE** para cada ocorrência.

É possível observar este exemplo nas duas primeiras linhas e após a quinta linha do *Preview Results* demonstrado acima.

Preview Results

STATE	CUSTOMERNAME	QUANTITYORDER...	PRICEEACH	PRODUCTLINE	EXTENDED_PRICE
CA	Technics Stores Inc.	27	57.32	Planes	1547.64
CA	Technics Stores Inc.	48	68.1	Planes	3268.7999999999997
CA	Technics Stores Inc.	28	89.9	Motorcycles	2517.2000000000003
CA	Technics Stores Inc.	31	57.78	Vintage Cars	1791.18
CA	Technics Stores Inc.	48	39.71	Vintage Cars	1906.08
CA	Technics Stores Inc.	28	91.34	Planes	2557.52
CA	Technics Stores Inc.	31	87.75	Planes	2720.25
CA	Technics Stores Inc.	36	94.92	Planes	3417.12
CA	Technics Stores Inc.	48	72.0	Planes	3456.0
CA	Technics Stores Inc.	39	67.37	Planes	2627.4300000000003
CA	Technics Stores Inc.	35	69.55	Planes	2434.25
CA	Technics Stores Inc.	22	182.04	Motorcycles	4004.8799999999997
CA	Technics Stores Inc.	22	131.04	Motorcycles	2882.8799999999997
CA	Technics Stores Inc.	23	53.91	Motorcycles	1239.9299999999998
CA	Technics Stores Inc.	50	91.29	Motorcycles	4564.5
MA	Cambridge Collect...	29	214.3	Classic Cars	6214.700000000001
MA	Cambridge Collect...	32	100.34	Trucks and Buses	3210.88
MA	Cambridge Collect...	24	101.31	Vintage Cars	2431.44
MA	Cambridge Collect...	45	57.46	Vintage Cars	2585.7
MA	Cambridge Collect...	31	100.53	Vintage Cars	3116.43
MA	Cambridge Collect...	33	84.73	Vintage Cars	2796.09
MA	Cambridge Collect...	46	88.93	Vintage Cars	4090.78
MA	Cambridge Collect...	20	54.81	Planes	1096.2
MA	Cambridge Collect...	25	65.75	Vintage Cars	1643.75
MA	Cambridge Collect...	40	85.99	Ships	3439.6
MA	Cambridge Collect...	32	49.16	Planes	1573.12
CT	Gift Depot Inc.	42	109.51	Classic Cars	4599.42
CT	Gift Depot Inc.	39	117.48	Classic Cars	4581.72
CT	Gift Depot Inc.	48	139.87	Classic Cars	6713.76
CT	Gift Depot Inc.	32	61.0	Classic Cars	1952.0
CT	Gift Depot Inc.	34	43.27	Classic Cars	1471.18
CT	Gift Depot Inc.	22	79.97	Classic Cars	1759.34
CT	Gift Depot Inc.	24	77.91	Classic Cars	1869.84
CT	Gift Depot Inc.	22	87.81	Classic Cars	1931.8200000000002
CT	Gift Depot Inc.	34	89.0	Motorcycles	3026.0
CT	Gift Depot Inc.	40	107.05	Motorcycles	4282.0
CT	Gift Depot Inc.	41	193.66	Motorcycles	7940.0599999999995
CT	Gift Depot Inc.	48	123.51	Motorcycles	5928.4800000000005
CT	Gift Depot Inc.	33	67.58	Planes	2230.14
CT	Gift Depot Inc.	34	50.27	Motorcycles	1709.18

Figura 32

11.1 CRIANDO UM *DATA CUBE*

Como explicado acima, o *Data Cube* é o componente que fornece os dados para um *Cross Tab*. Sendo assim, ele precisa que um *Data Set* forneça uma tabela de resultados onde o *Data Cube* irá preparar os “dados cruzados”.

O *Data Cube* trata um conjunto de resultados como uma dimensão. Portanto, para o funcionamento de um *Cross Tab*, é preciso no mínimo duas dimensões.

É possível trabalhar com várias dimensões e as interseções destes dados serão exibidos nas células do *Cross Tab*.

Considerando que já exista um *Data Set*, siga os passos a seguir para a criação do *Data Cube*.

As imagens foram extraídas do site oficial do Eclipse.

1. No **"Data Explorer"**, clique com o botão direito em **Data Cubes** e selecione a opção **novo**.
2. Escolha um nome para o *Data Cube* no campo **Name**.
3. Selecione o *Data Set* principal em **Primary dataset**.
4. Caso precise trabalhar com mais de um *Data Set* e houver agregação nos dados de outros *Data Sets*, marque a opção **Check this option to avoid preaggregation in primary data set during cube generation**.
5. Selecione **Groups and Summaries** na árvore localizada à esquerda;
6. Para criar as dimensões, arraste os campos da árvore em **Available Fields** para o item **Groups (dimensions)** em **Groups and Summaries**.
7. Será solicitado um novo nome para o grupo.

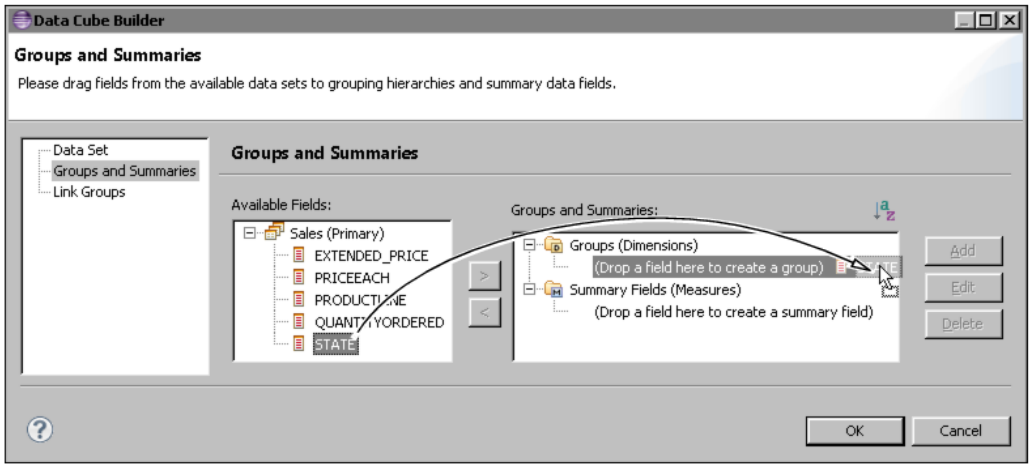


Figura 33

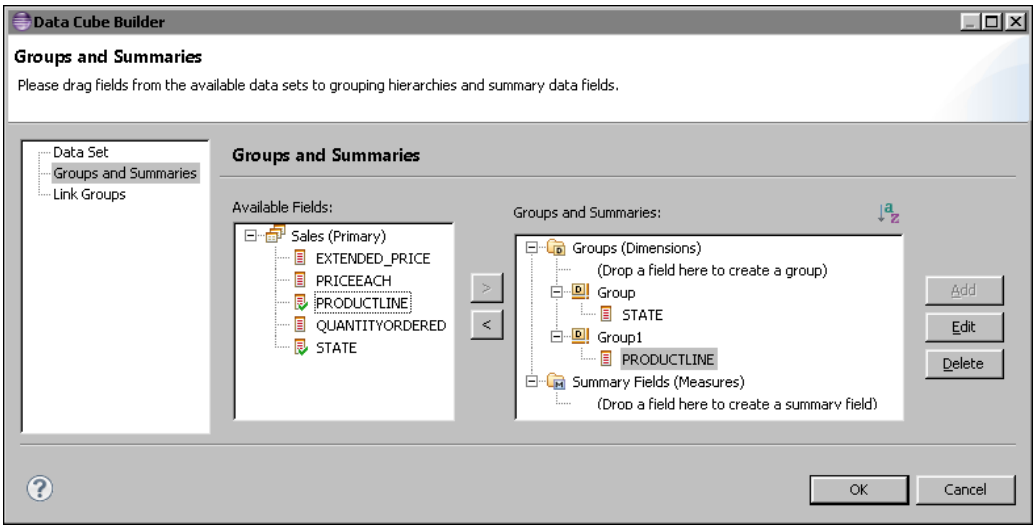


Figura 34

8. Para os resultados (agregados) que serão exibidos nas interseções, o procedimento é semelhante item acima. Entretando, arraste o campo para **Summaries Fields (Measures)**;

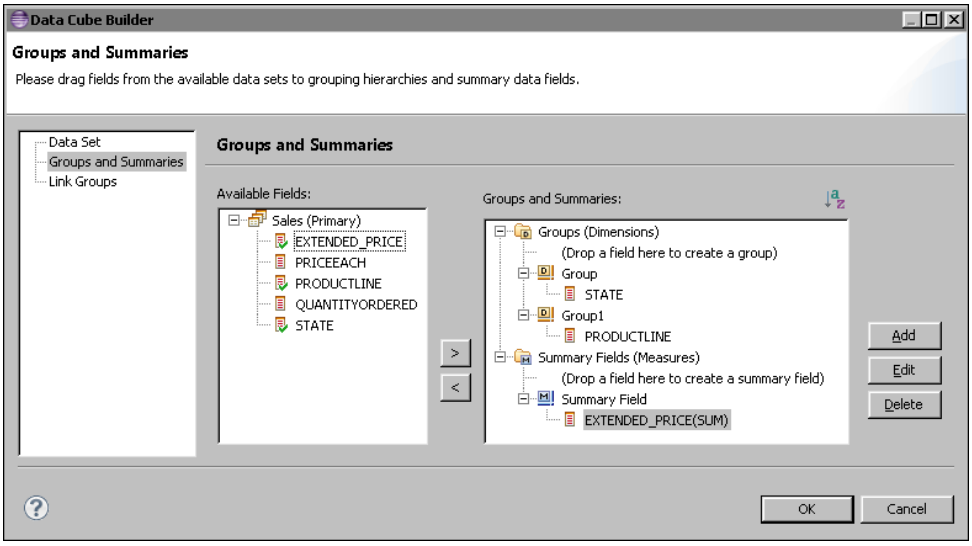


Figura 35

9. Clique no botão **OK**.

11.2 CRIANDO UM CROSS TAB

Com o *Data Cube* criado, siga os passos abaixo:

1. Inicie inserindo um *Cross Tab* através de duas formas:

   - Clique com o botão direito no local desejado no editor, selecionar a opção **Insert** e, em seguida, **Cross Tab**.
   - Clique e arraste o *Data Cube* desejado da árvore do Data Explorer diretamente para a área do relatório.

2. Expanda os componentes do *Data Cube* em questão na árvore em **Data Explorer**.

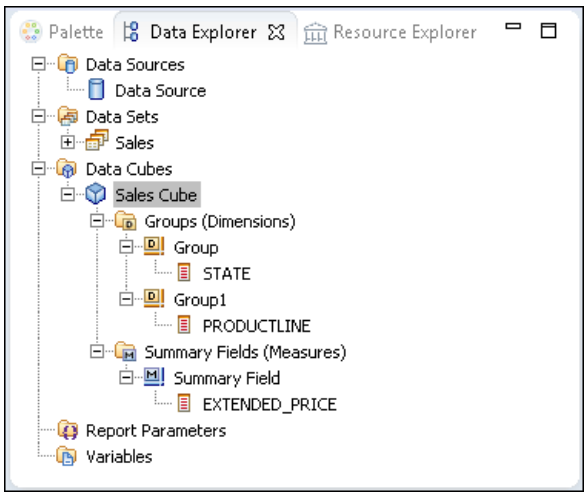


Figura 36

3. Para definir a coluna esquerda, arraste o(s) campo(s) do *Data Cube* para a área escrita **Drop data field(s) to define rows here**.
4. Para definir a linha superior, arraste o(s) campo(s) do *Data Cube* para a área escrita **Drop data field(s) to define columns here**.
5. Para os resultados, arraste o(s) campo(s) do *Data Cube* para a área escrita **Drop data field(s) to be summarized here**.

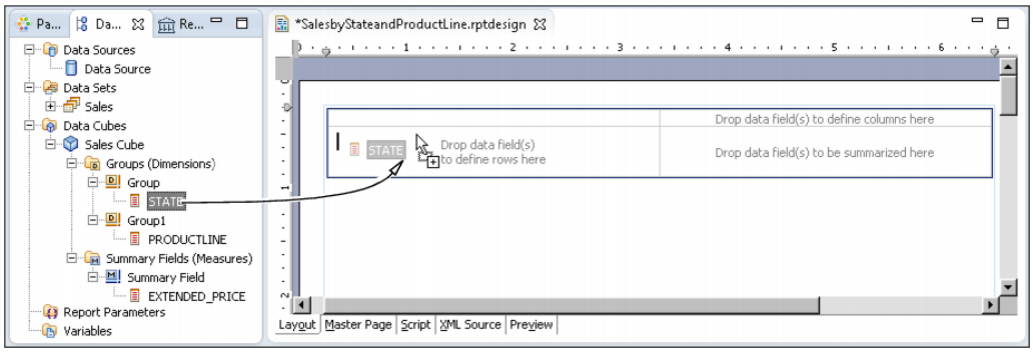


Figura 37

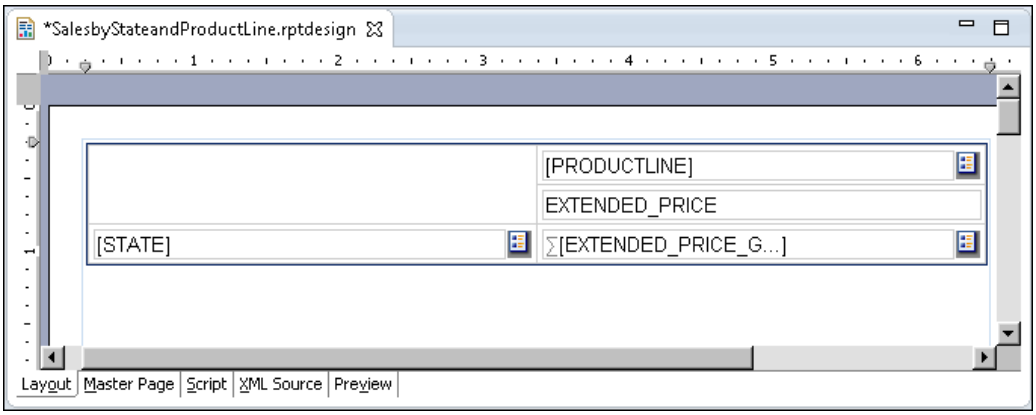


Figura 38

Integração

O material abaixo é baseado na documentação oficial sobre o BIRT Report Viewer do Eclipse: `www.eclipse.org/birt/phoenix/deploy/viewerUsage.php`.

Para mais informações de integração do BIRT a outras aplicações, consulte o endereço: `www.eclipse.org/birt/phoenix/deploy/`.

Para visualizar um relatório em uma aplicação web, utilize o seguinte formato de URL:

```
http://sitePrefix/birt-viewer/run?option1=value1&option2=value2&...
```

A organização do endereço segue o seguinte padrão:

- <sitePrefix> é a URL base (incluindo a porta) da instalação do servidor de aplicação.
- Os parâmetros (option1, option2, etc.) são utilizados para transferir a localização e o nome do arquivo do relatório, as configurações de renderização e os parâmetros definidos pelo usuário (se houver) ao TOMcat.

A ordem em que os parâmetros são informados na URL não é relevante.

Observe abaixo a lista dos principais parâmetros de configuração do relatório:

Parâmetro	Descrição	Valores	Default	frameset	run
__format	Formato de saída	html ou pdf	html	Não	Sim
__isnull	Identifica que um parâmetro de relatório tem um valor nulo	Nome do parâmetro	Nenhum. Requerido.	Não	Sim
__locale	Local, região	Região em Java, ex.: en, pt-br, en-us ou ch-zh	JVM locale	Sim	Sim
__report	Caminho e nome do relatório		Nenhum. Requerido.	Sim	Sim
__document	Caminho e nome do documento (relatório)		Nenhum. Requerido.	Sim	Não
Parâmetro do relatório	Parâmetros do relatório	Nome e valores dos parâmetros do relatório	De acordo com o projeto do relatório	Sim	Sim

- **Framset:** Apresenta um conjunto de framset que contém o relatório e componentes adicionais AJAX, entrada de parâmetros e controles de paginação.
- **Run:** Executa o relatório e exibe a saída como uma página HTML autônomo, ou como um documento PDF.

Observe o exemplo abaixo retirado do site Eclipse:

```
http://localhost:8080/birt-viewer/run__report=report%5CSalesInvoice.rptdesign&
OrderNumber=10010
```

- **http:localhost:8080**: A URL (incluindo a porta) da instalação do *app server*.
- **birt-viewer**: O nome do *BIRT Viewer servlet*.
- **report=report%5CSalesInvoice.rptdesign**: Caminho relativo do local do arquivo. O caminho é relativo ao diretório *web app* em que o *birt-viewer* foi instalado.
- **OrderNumber=10010**: O nome e valor de um parâmetro de relatório, conforme definido no projeto de relatório.

12.1 EXEMPLO DE INTEGRAÇÃO

12.1.1 *URL* do Relatório

O exemplo abaixo é baseado no produto Folha, sendo uma URL típica de uma chamada de relatório. Neste exemplo, o relatório Relação de Tomadores (tomadores.rptdesign):

```
http://saas.teknisa.com:9191/birt-viewer/frameset?__report=%2Fvar%2Fwww%2Fhtml%2F
teknisa_saas%2Fhomologacao%2Fv_220%2Ffpa-teknisa%2Fproducts%2Freports%2Fbirt%2Ffpa
%2Ftomadores.rptdesign&__format=pdf&__svg=false&__locale=pt_BR&__timezone=America2
FSao_Paulo&P_VERSAO=5.07.006+WEB&P_NRORG=1001&P_COMPETENCIA=01/10/2013&P_VINCULO=&
P_ESTRUTLEGAL=&P_ESTRUTGEREN=&P_INCLUIRDEMITIDO=S&ZEBRADO=S&P_CONFIDENCIAL=VH.CONF
IDENCIAL%20=%20'N'&CDOPERADOR=000000002848&NMOPERADOR=Rafael%20Pickbrenner&IMG_SRC
=&PATH=/var/www/html/teknisa_saas/homologacao/v_220/fpa-teknisa/products/projects
/fpa/config/config.ini
```

Observe a organização da URL:

Parâmetro da URL	Descrição
`http://saas.teknisa.com:9191/birt-viewer/frameset`	A URL (incluindo a porta) da instalação do *app server*.
`?__report=%2Fvar%2F...%2Ftomadores.rptdesign`	Caminho relativo do relatório.
`&__format=pdf`	Parâmetros de configuração do relatório.
`&__svg=false`	

Parâmetro da URL	Descrição
&__locale=pt_BR	
&__timezone=America%2FSao_Paulo	
&P_VERSAO=5.07.006+WEB	Parâmetros do relatório.
&P_NRORG=1001	
&P_COMPETENCIA=01/10/2013	
&P_VINCULO=	
&P_ESTRUTLEGAL=	
&P_ESTRUTGEREN=	
&P_INCLUIRDEMITIDO=S	
&ZEBRADO=S	
&P_CONFIDENCIAL=VH.CONFIDENCIAL%20=%20'N'	
&CDOPERADOR=000000002848	
&NMOPERADOR=Rafael%20Pickbrenner	
&IMG_SRC=	Local de imagens.
&PATH=/var/www/.../config.ini	Configurações adicionais.

- **Parâmetros referentes a configurações de estilo para o relatório:**
  - __report
  - __format
  - __svg
  - __locale
  - __timezone

- **Parâmetros de definição aplicadas pelo o usuário (e/ou sistema):**
  - P_VERSAO
  - P_NRORG
  - P_COMPETENCIA
  - P_VINCULO
  - P_ESTRUTGEREN
  - P_INCLUIRDEMITIDO
  - ZEBRADO
  - P_CONFIDENCIAL
  - CDOPERADOR
  - NMOPERADOR
  - IMG_SRC

12.1.2 Código PHP

A seguir serão apresentados alguns dos códigos usados no produto **Folha** para o tratamento dos relatórios.

1. O método **fichaFinanceiraGrade** recupera os dados (parâmetros) do ambiente, verifica a validade de alguns destes dados, prepara os dados e então executa o relatório através do método **tk_open_report_from_array()** (ver figura 39).

2. O método **tk_open_report_from_array()** verifica o tipo de relatório e executa o método necessário, neste caso o **tk_open_birt_report_from_array()** que está detalhado na figura.
3. O **tk_open_birt_report_from_array()** executa o relatório. Para isso são utilizados três métodos: **setReport()**, **setParameterValuesFromArray()** e **viewReport()** para preparar a URL e executar o relatório (ver figura 40).
4. O **setReport()** faz uma preparação prévia na *string* que irá conter a URL do relatório, e o **setParameterValuesFromArray()** finaliza a montagem da *string* da URL (ver figura 41).

   Como mencionado anteriormente, o “viewReport()” é encarregado de exibir o relatório em uma nova janela.

```php
/*
 * Geta para um array os parametros necessários para o relatorio "Ficha Financeira Grade"
 * e faz a chamada do relatório correspondênte.
 */
public static function fichaFinanceiraGrade($nmreport) {
    $mesComp          = tk_get_field_value('MESCOMP');
    $nrOrg            = tk_get_field_value('NRORG');
    $nomeVinculo      = tk_get_field_value('NOMEVINCULOATIVO');
    $nrEstrutura      = tk_get_field_value('NRESTRUTURA');
    $nrEstruturaGeren = tk_get_field_value('NRESTRUTGERENCIAL');
    $nrTpModalidCal   = tk_get_field_value('NRTPMODALIDCAL');
    $tipoMovimento    = tk_get_field_value('TIPOMOVIMENTO');
    $NrEvento         = tk_get_field_value('NREVENTO');
    $DataInicial      = tk_get_field_value('DATAINICIAL');
    $DataFinal        = tk_get_field_value('DATAFINAL');
    $zebrado          = tk_get_field_value('ZEBRADO');
    $IncluirDemitidos = tk_get_field_value('INCLUIRDEMITIDOS');

    if($nrTpModalidCal == null || $tipoMovimento == null || $DataInicial == null || $DataFinal == null ) {
        tk_warning('Os campos Modalidade de Cálculo, Tipo de Movimento e Data Inicial e Final devem ser
            preenchidos para a emissão do relatório Ficha Analítica.');
    }else{
        //*Inicia o array que sera enviado para o relatorio
        $params = array();
        $params['P_COMPETENCIA']     = $mesComp;
        $params['P_NRORG']           = $nrOrg;
        $params['P_VINCULO']         = $nomeVinculo;
        $params['P_ESTRUTLEGAL']     = $nrEstrutura;
        $params['P_ESTRUTGEREN']     = $nrEstruturaGeren;
        $params['P_NRTPMODALIDCAL']  = $nrTpModalidCal;
        $params['P_NRTPMOVIMENT']    = $tipoMovimento;
        $params['P_NREVENTO']        = $NrEvento;
        $params['P_INICIO']          = $DataInicial;
        $params['P_FIM']             = $DataFinal;
        $params['ZEBRADO']           = $zebrado;
        $params['P_INCLUIRDEMITIDO'] = $IncluirDemitidos;
        $params['P_OCORRECAL']       = \FPA::getOcorrenciaCalculo();
        //Parâmetro de confidencialidade
        $params['P_CONFIDENCIAL']    = self::setQueryConfidencial(tk_get_field_value('SELECAOCONF'));
        $params['CDOPERADOR']        = \Ambiente::getCdOperador(); //Dados do operador
        $params['NMOPERADOR']        = \Ambiente::getNmOperador();
        tk_open_report_from_array($nmreport, $params, \Setforms2\Runtime\Helper\Report\TkBirtReport::FORMAT_PDF);
    }
}
```

Figura 39

```php
/**
 *
 * @param type $reportName
 * @param array $params
 * @param type $format
 * @param type $mode
 * @param string $reportType birt|qr2
 */
function tk_open_report_from_array($reportName,
        array $params,
        $format = \Setforms2\Runtime\Helper\Report\TkBirtReport::FORMAT_HTML,
        $mode = 'frameset',
        $reportType = 'birt',
        $targetFileName = null){
    if ($reportType === 'birt')
        tk_open_birt_report_from_array ($reportName, $params, $format, $mode);
    else if ($reportType === 'qr2')
        tk_open_qr2_report_from_array($reportName, $params, $targetFileName);
}

function tk_open_birt_report_from_array($reportName,
        array $params,
        $format = \Setforms2\Runtime\Helper\Report\TkBirtReport::FORMAT_HTML,
        $mode = 'frameset'){
    $birt = \Setforms2\Runtime\Helper\Report\TkBirtReport::getInstance();
  1 $birt->setReport($reportName, 'E');
  2 $birt->setParameterValuesFromArray($params, $format);
  3 $birt->viewReport($mode);
}
```

Figura 40

```
1  public function setReport($reportName, $type)
24     {
25         $this->_type = $type;
26         $reportPath = $this->getReportPath($this->_type);
27         $this->_report_path = $reportPath;
28         $this->_report = $this->_report_path . DIRECTORY_SEPARATOR . $reportName . '.rptdesign';
29         parent::setReport($this->_report);
30     }

54  2 public function setParameterValuesFromArray(array $ext_params, $format)
55     {
56         foreach ($ext_params as $param_name => $value) {
57             $params[] = "$param_name=$value";
58         }
59
60         $params_str = '&'.implode('&', $params);
61         $this->setParameterValues($params_str, $format);
62     }

       public function setParameterValues($params = '', $format = self::FORMAT_HTML)
46     {
47         $this->_params = "&__format={$format}&__svg=false&__locale=pt_BR&__timezone=America%2FSao_Paulo";
48         $this->_params .= "&P_VERSAO=" . urlencode($this->_version);
49         $this->_params .= $params;
50         $this->_params .= "&IMG_SRC=" . urlencode($this->getLogotype());
51         parent::setParameterValues($this->_params);
52     }

3  public function viewReport($mode = 'frameset')
65  {
66      parent::viewReport($mode);
67  }

       protected function viewReport($mode)
39     {
40         $url = $this->getUrl($mode);
41         tk_open_report_in_new_window(base64_encode(tk_curl_download($url)));
42     }

542 function tk_open_report_in_new_window($data){
543     tk_add_method('redirectToUrl', array("data:application/pdf;".$data, true));
544 }
```

...

Figura 41

Tomcat

O **Apache Tomcat** é um servidor de aplicações *Java* (*Contêiner de Servlets*) utilizado para executar aplicações *web* escritas em *Java*. Para sua utilização, é necessário ter o **JRE** (*Java Runtime Environment*) instalado no sistema.

O Tomcat atua como o hospedeiro para o visualizador *web* do **BIRT**, que por sua vez contém o motor de *renderização* necessário para exibir os relatórios.

A seguir, será detalhada a instalação do Tomcat e do visualizador do BIRT.

Instalação do Apache Tomcat

1. Realize o *download* do instalador do Tomcat (`tomcat.apache.org/download-60.cgi`). Recomenda-se o arquivo *"Windows Service Installer"* da seção *"Binary Distributions"*.
2. Anote o caminho da instalação do Tomcat para uso posterior.
3. A porta padrão do Tomcat é a **8080**, mas pode ser alterada. Defina um nome de usuário e senha para o acesso.
4. Na tela final, marque la opção para executar o Tomcat e conclua a instalação.
5. É necessário criar uma variável de ambiente, mais precisamente uma variável do sistema. Crie uma variável de ambiente do sistema chamada **CATALINA_HOME**, cujo valor deve ser o caminho da instalação do Tomcat.
6. Configure como o serviço será executado, inicie-o e clique em **"OK"**. Para verificar, acesse `http://localhost:8080` (ou a porta configurada) em seu navegador. A página de boas-vindas do Apache Tomcat deve aparecer.

Instalação do BIRT Viewer

1. No arquivo compactado do BIRT, localize a pasta **"Web Viewer Example"**. Se não o tiver, baixe o *"BIRT report engine runtime"* do site do Eclipse.
2. Renomeie a pasta **"Web Viewer Example"** para **birt-viewer**.
3. Copie a pasta **birt-viewer** para dentro da pasta **webapps** no diretório de instalação do Tomcat.
4. Reinicie o serviço do Tomcat.
5. No navegador, na página do Tomcat, clique no botão **"Manager App"**. Insira o usuário e a senha definidos na instalação do Tomcat, se solicitado.
6. Na nova janela, a aplicação /**birt-viewer** deverá estar listada. Verifique seu status e clique sobre seu nome.
7. Uma página de boas-vindas do Eclipse deverá aparecer. Clique no *link* **"View Example"**.
8. Se uma mensagem de sucesso for exibida, o *BIRT Viewer* está funcionando corretamente.
9. Por padrão, os arquivos de relatório (**.rptdesign**) devem ser colocados na pasta **webapps/birt-viewer/** da instalação do Tomcat. É possível que os arquivos estejam em outras pastas, desde que o caminho completo seja informado na variável da URL.

Notas

Este tópico lista algumas observações e sugestões de prática em relação ao **BIRT**.

- **Parâmetros *null* via PHP:** Ao passar o valor *null* para um parâmetro via PHP, o BIRT o interpreta como uma *string* vazia (), e não como *null*.
- **Depuração de Erros de *Query*:** Em caso de erro na execução da *query*, procure pela seção no *log* de erro que inicia com: **Caused by: org.eclipse.birt.data.engine.odaconsumer.OdaDataException: ...**. A descrição mais precisa do erro geralmente se encontra no final desta linha.
- **Consistência nos Parâmetros:** Mantenha uma lógica consistente ao utilizar parâmetros em diferentes partes do relatório. Por exemplo, se um parâmetro **INCLUIR_DEMITIDOS** utiliza o valor **'S'** para ativar uma funcionalidade, toda a lógica condicional no relatório (seja na *query* ou na visibilidade de um componente) deve verificar apenas a presença ou ausência do valor **'S'**, tratando-o como o único gatilho válido.
- **Visibilidade (*Visibility*):** Lembre-se que, na propriedade *"Expression"*, o retorno **true** torna o elemento invisível, enquanto **false** o mantém visível.
- **Filtros e Agregações:** Filtros aplicados pelo *Filters* devem incluir a mesma condição em *Aggregations*, como totalizadores. Por isso, é recomendado que as *queries* contenham o filtro embutido.
- **Parâmetros de Data:** As funções de data nativas do BIRT podem não funcionar corretamente com parâmetros do tipo *String* que contêm uma data. A prática recomendada é passar a data como um parâmetro do tipo *String*, incluí-lo como um campo no *dataset* e, então, aplicar as funções de data do BIRT sobre este campo do *dataset*.
- **Cláusula IN com Parâmetros:** Não é possível passar uma lista de valores (ex: "1,2,3") para um único parâmetro e utilizá-la diretamente em uma cláusula **IN** na *query*. A solução é construir a *query* dinamicamente via *script*, substituindo um marcador no texto da *query* pelo valor completo do parâmetro.
- **Uso do Desfazer (Ctrl+Z):** Ao reverter uma alteração, prefira utilizar o comando **Desfazer (Ctrl+Z)** em vez de reconfigurar manualmente. O BIRT pode, em casos raros, apresentar instabilidade se as propriedades forem alteradas manualmente fora da sequência esperada.
- **Tratamento de *null* em *Scripts*:** *Scripts* em BIRT interpretam um parâmetro com valor *null* como uma *string* vazia (). É importante notar que esta conversão se aplica apenas a parâmetros; valores *null* retornados de um campo do *dataset* são corretamente identificados como *null* nos *scripts*.

A seguir, serão apresentadas algumas sugestões de práticas de código **SQL** para que a *query* possa trabalhar com parâmetros opcionais (parâmetros que podem ou não ser preenchidos). Isto confere maior flexibilidade à *query*.

Por exemplo, um usuário pode desejar o histórico completo de movimentação de estoque (deixando as datas em branco) ou pode informar uma ou ambas as datas para refinar a busca.

Os exemplos a seguir ilustram situações onde o parâmetro é opcional e, quando informado, a *query* considera apenas este parâmetro.

```
AND ( VM.DTRESCISAOVINC > LAST_DAY(:P_COMPETENCIA) OR VM.DTRESCISAOVINC IS NULL
OR :P_INCLUIRDEMITIDO = 'S' )

AND ( ICF.NREVENTOM = 3030 OR :P_SOMENTELIQUIDO = 'S' ) --Filtro do evento
```

O exemplo abaixo ilustra um parâmetro que pode ser utilizado para **determinar se uma condição será atendida**:

- **Exemplo 1:** Um parâmetro define se funcionários rescindidos devem ou não ser incluídos nos resultados.
- **Exemplo 2:** Um parâmetro define se a consulta deve retornar todos os eventos ou apenas os registros de um evento específico (ex: evento '3030').


```
AND ( VM.DTRESCISAOVINC > LAST_DAY(:P_COMPETENCIA) OR VM.DTRESCISAOVINC IS NULL
OR :P_INCLUIRDEMITIDO = 'S' )

AND ( ICF.NREVENTOM = 3030 OR :P_SOMENTELIQUIDO = 'S' ) --Filtro do evento
```


O BIRT também aceita *scripts* em vários de seus componentes.

O *script* abaixo substitui o texto **:REPORT_CONDITION** por um texto de um parâmetro.

```
//script no evento "beforeOpen" nos Data Sets do birt
//(APENAS NOS QUE USARAM O "AND :REPORT_CONDITION" na(s) query(s)).
//Deve estar em todos os Data Sets no qual se precisa confidencializar alguma
informação

if(params["P_CONFIDENCIAL"].value == "" || params["P_CONFIDENCIAL"].value == null)
    //Verifica se está recebendo um fragmento de query
{
    //Não houve um fragmento de query. Tratando para não gerar erro.
    this.queryText = this.queryText.replaceAll(":REPORT_CONDITION", "1 = 1");
} else {
    //Entrada de fragmento de query. Adicionado à query original do birt.
    this.queryText = this.queryText.replaceAll(":REPORT_CONDITION",
    params["P_CONFIDENCIAL"].value);
}
```

O script abaixo altera a **orientação da página para paisagem**:

```
// Para alterar a orientação da página
// NO EVENTO beforeFactory
reportContext.getDesignHandle().findMasterPage("Simple MasterPage").orientation =
"Landscape";
```