Modulo III – Relatórios e Gráficos em Java

Prof. Ismael H F Santos

Ementa

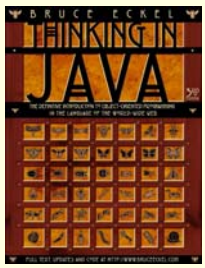
- Modulo III – Relatórios em JAVA
  - Relatórios - Jasper e IReports
- Modulo III – Gráficos em JAVA

Bibliografia

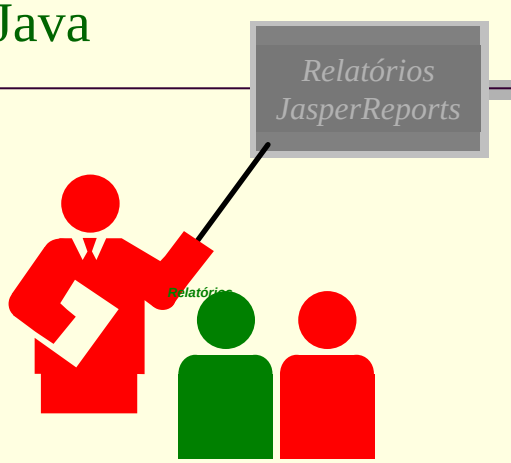
- *Linguagem de Programação JAVA*
  - *Ismael H. F. Santos, Apostila UniverCidade, 2002*
- *The Java Tutorial: A practical guide for programmers*
  - *Tutorial on-line: http://java.sun.com/docs/books/tutorial*
- *Java in a Nutshell*
  - *David Flanagan, O´Reilly & Associates*
- *Just Java 2*
  - *Mark C. Chan, Steven W. Griffith e Anthony F. Iasi, Makron Books.*
- *Java 1.2*
  - *Laura Lemay & Rogers Cadenhead, Editora Campos*

Livros

- **Core Java 2**, Cay S. Horstmann, Gary Cornell
  - Volume 1 (Fundamentos)
  - Volume 2 (Características Avançadas)
- **Java: Como Programar**, Deitel & Deitel
- **Thinking in Patterns with JAVA**, Bruce Eckel
  - **Gratuito**. http://www.mindview.net/Books/TIJ/



FPSW-Java



JasperReports

- Framework open-source escrito em Java para geração de relatórios

- http://jasperreports.sourceforge.net/

iReport

- O iReport é uma ferramenta que visa facilitar a construção de relatórios (layout) utilizando a biblioteca JasperReports através de uma interface gráfica desenvolvida em Swing.
- Fornece suporte à construção de relatórios complexos.
  - Elimina a necessidade de manipulação direta dos arquivos JRXML.

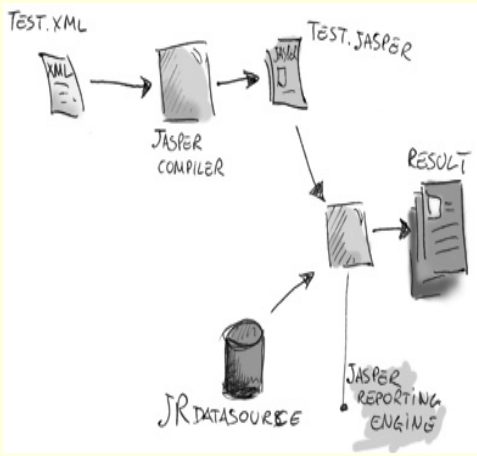
Usando JasperReports/iReport

1. Criação do desenho (layout) do relatório;
2. Preenchimento do relatório com dados;
3. Visualização (e/ou exportação) do relatório (PDF, HTML, XLS, etc).

- Obs.:
  - passo 1 é realizado no **iReport**;
  - passos 2 e 3 são programáticos, com o uso da biblioteca **JasperReports**.

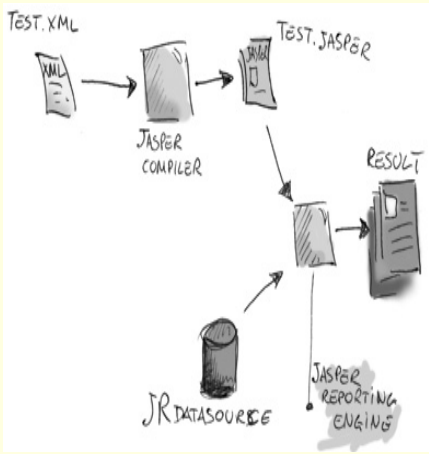
Funcionamento

- O design do relatório é definido em um arquivo XML, que obedece a estrutura declarada no arquivo *jasperreports.dtd*.
- O arquivo XML é compilado gerando um arquivo **.jasper**, onde as expressões Java existentes dentro do XML serão verificadas em tempo de compilação.



Funcionamento (cont.)

- Pode-se ainda definir os campos que serão preenchidos dinamicamente a partir de uma base de dados.
- Diferentes objetos Jasperreports são usados para representar as etapas do processo de geração de relatórios:
  - **JasperDesign** <- test.xml
  - **JasperReport** -> test.jasper
  - **JasperPrint** -> test.pdf



JasperReports DataSources

- Para produzir um relatório precisamos fornecer dados ao Jasper.
  - consultas SQL inserida no arquivo XML
  - um objeto ResulSet gerando por uma classe Java
- Interface JRDataSource abstrai diferentes fontes de dados, fornecendo os seguintes tipos:
  - JREmptyDatasource - especial datasource usado para preencher relatórios que não possuem registros ou dados recuperados
  - JRResultSetDataSource - implementação padrão desta interface para objetos ResultSet
  - XML DataSource - empacotar um arquivo XML e normalizar seu conteúdo. As únicas informações necessárias para criar este tipo de datasource são: o nome do datasource e o nome do arquivo XML.

JasperReports DataSources

- TableModel DataSource – acessa tabelas já carregadas em interfaces swing.
- JavaBeans Set Datasource - capaz de empacotar uma Collection ou um Array de JavaBeans. É necessário uma classe especial de fábrica (factory) que forneça um método estático para gerar a coleção ou um array de JavaBeans.
- Custom Datasource - datasource genérico. É necessário uma classe especial de fábrica (factory) que forneça um método estático que retorne um JRDataSource.

- No linguajar “Jasper", um datasource somado a um arquivo .jasper gera um "print", que pode ser "exportado" para os formatos PDF, HTML, XML, CVS ou XLS.

Campos e Parâmetros

- ***Campos*** (Fields) são "áreas específicas" no relatório que receberão diretamente os dados das respectivas colunas referenciadas.

  <field name="Nome" class="java.lang.String"/>
- ***Parâmetros*** são dados passados para a operação de preenchimento, que não podem ser encontrados normalmente na fonte de dados.

  <parameter name="TituloDoRelatorio" class="java.lang.String"/>
- Passados via código Java, através da classe HashMap:

  Map parametros = new HashMap( ); parametros.put("Cliente", "Fulano de Tal");
- Utilizados, por exemplo, na query do relatório

  Select * FROM CLIENTE WHERE CLIENTE=$P{Cliente}

Variáveis

- ***Variáveis*** são utilizadas para armazenar resultados temporários necessários para geração do relatório
  - podem referenciar tipos internos de cálculos, como contagem (count), soma (sum), média (average), menor (lowest), maior (highest), etc

<variable name="ValorTotalCompraSum" class="java.lang.Double" calculation="Sum">
<variable expression> ${ValorProduto} </variable expression>
</variable>

Variáveis

- A ordem em que as variáveis são declaradas no relatório é importante.
- Podemos definir o nível no qual uma variável irá ser inicializada. Pode ser no início do relatório (uma única vez), a cada página, coluna ou grupo.

```
<variable name="ValorTotalCompraSum"
   class="java.lang.Double" resetType="Page"
   calculation="Sum"> <variable expression>
   ${ValorProduto} </variable expression>
   <initialValueExpression> new Double( 0 )
   </initialValueExpression>
</variable>
```

- Variáveis internas da ferramenta: PAGE_NUMBER, COLUMN_NUMBER, REPORT_COUNT, PAGE_COUNT, COLUMN_COUNT.

Expressões

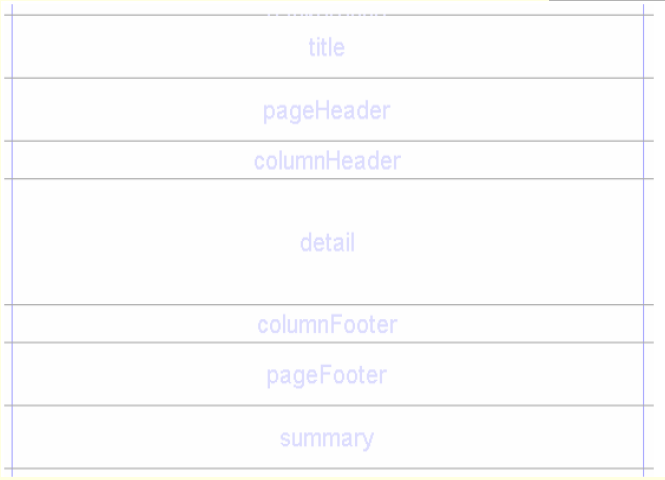
- ***Expressões*** (Expressions) são utilizadas para especificar o conteúdo de campos de texto, na realização de cálculos freqüentes
- Todas elas são expressões Java que podem conter em sua sintaxe:
  - campos: acessado com $F{nome}
  - parâmetros: acessado com $P{nome}
  - variáveis de relatório: acessado com $V{nome}.
- Exemplo de uma expressão:

```
<textFieldExpression> "Sr.(a) " + $F{Cliente} + " realizou um
   total de compras no valor de " + $V{ValorTotalCompraSum}
   + " no dia " + (new SimpleDateFormat("dd/MM/yyyy"))
   .format($F{DataCompra}) + "."
</textFieldExpression>
```

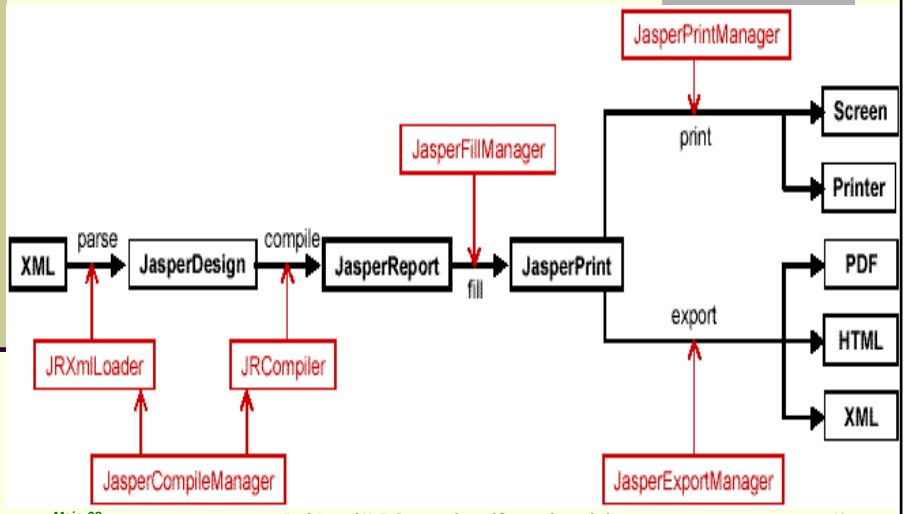
Layout

- O JasperReports divide o layout do relatório em áreas “pré-definidas”, chamadas seções.
- As seções levam em considerção a estrutura visual de um relatório. São elas: **background**, **title**, **pageHeader**, **columnHeader**, **detail**, **columnFoter**, **pageFooter**, **lastPageFooter** e **summary**.

Seções do layout do relatório



JasperReport API



JasperReport API (cont.)

- Classe net.sf.jasperreports.engine.design.JasperDesign
  - representam o relatório no seu formato mais primitivo. São resultados de um processamento sobre o arquivo XML.
- Classe net.sf.jasperreports.engine.JasperReport
  - relatórios compilados armazenados em arquivos *.jasper*. Nesse estágio, toda a análise sintática nas expressões existentes no XML já foram realizadas.
- Classe net.sf.jasperreports.engine.JasperPrint
  - JasperReport com todos os campos preenchidos, pode ser visualizado diretamente utilizando visualizadores internos do JasperReport. Também pode ser transformado em formatos mais populares como HTML, XML ou PDF

JasperReport API (cont.)

- Interface net.sf.jasperreports.engine.JRDataSource
  - padroniza o comportamento das classes que manipulam as fontes de dados necessárias durante o preenchimento dos campos existentes no JasperReport.
- Classe net.sf.jasperreports.engine.JasperFillManager
  - utilizada para gerar instâncias da classe JasperPrint utilizando uma fonte de dados (JRDataSource) e uma instância da classe JasperReport.
- Classe net.sf.jasperreports.engine.JasperPrintManager
  - Permite imprimir o relatório completo ou páginas do mesmo. É possível imprimir o relatório como uma imagem (utilizando o método printPageToImage)

JasperReport API (cont.)

- Classe net.sf.jasperreports.engine.JasperExportManager
  - gera documentos nos formatos PDF, HTML e XML (v 1.0).
- Exemplo

```
import java.sql.Connection;
....
import net.sf.jasperreports.engine.xml.JRXmlLoader;
import net.sf.jasperreports.view.JasperViewer;
public class JasperReportExemple {
  private static final String url = "jdbc:mysql://127.0.0.1/teste";
  private static final String driver = "com.mysql.jdbc.Driver";
  private static final String login = "";
  private static final String pwd = "";
  public JasperReportExemple() { }
```

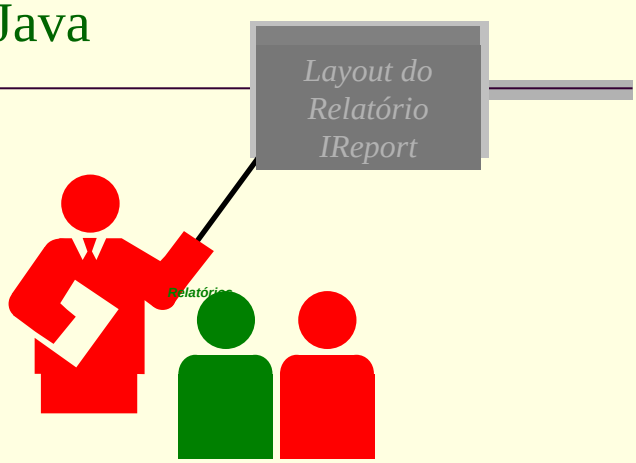
JasperReport API (cont.)

```
public void gerar( String layout ) throws JRException , SQLException,
    ClassNotFoundException {
     //gerando o jasper design JasperDesign
     desenho = JRXmlLoader.load( layout );
     //compila o relatório JasperReport
     relat = JasperCompileManager.compileReport( desenho );
     //estabelece conexão
     Class.forName( driver );
     Connection con = DriverManager.getConnection( url , login , pwd );
     Statement stm = con.createStatement();
     String query = "select * from turma";
     ResultSet rs = stm.executeQuery( query );
     //implementação da interface JR ResultSetDataSource
    JRResultSetDataSource jrRS = new JRResultSetDataSource( rs );
```


JasperReport API (cont.)

```
     //executa o relatório
     Map params = new HashMap();
     params.put("nota", new Double(10));
     JasperPrint imp = JasperFillManager.fillReport( relat , params, jrRS );
     //exibe o resultado
     JasperViewer viewer = new JasperViewer( imp , true );
     viewer.show();
}
public static void main(String[] args) {
   try { new JasperReportExemple().gerar( "report.jrxml" );
   } catch (Exception e) {  e.printStackTrace();   }
 }
 }
```

FPSW-Java

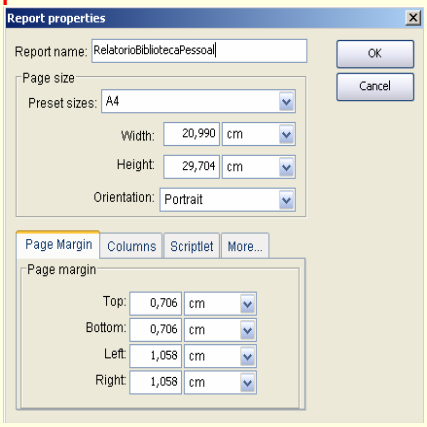


IReport

- O iReport é uma ferramenta que permite definir o design do relatório dentro de um ambiente gráfico, contento “todos” os recursos que a biblioteca Jasper oferece.
- É possível definir relatórios com designs modernos e complexos sem se quer escrever uma linha de código XML, que é todo gerado automaticamente

Criar um arquivo JRXML

- Relatório no iReport é armazenado em um .jrxml
  - No menu File, clique em New Document;
  - Aparece a janela para configuração do seu relatório:
    - o nome do relatório,
    - tamanho da folha,
    - margens,
    - colunas do relatório



Seções do layout do relatório

- Design do relatório utiliza a barra de ferramentas

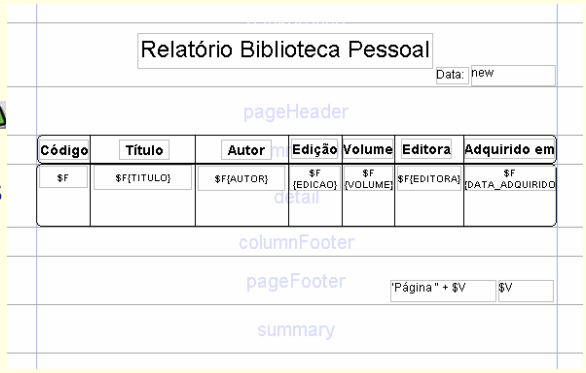
- O conteúdo da banda **title** aparece somente uma vez no começo do relatório.
- O que definimos na banda **pageHeader** aparece no alto de cada página do mesmo.
  - Esta parte pode, por exemplo, conter a data/hora e/ou o nome da organização.
- O **columnHeader** lista nomes daqueles campos específicos que você quer apresentar.
  - Por exemplo do “Nome do empregado”, “Hora de Início”, “Hora de término”, “Horas trabalhadas”, etc.

Seções do layout do relatório

- O **detail** é a banda onde os valores dos campos são apresentados. Por exemplo, “Jorge Horacio”, “12:00h”, “18:00h”, 06 horas”
- O **columnFooter** pode indicar a soma de alguns dos campos. Por exemplo “Horas totais trabalhadas: 180”
- O **pageFooter** aparece no final de cada página. Pode conter o número da página como “1/7”.
- O **summary** é a banda onde a informação inferida a partir dos dados da banda “detalhe” é indicada.
  - Por exemplo, após ter listado as horas trabalhadas para cada empregado na banda do “detalhe”, as horas totais trabalhadas para cada empregado podem ser apresentadas em um gráfico de torta.

Exemplo

- A tabela a seguir os textos “**Relatório Biblioteca Pessoal**”, “**Data**:”, “**Código**”, “**Título**”, “**Volume”, “Edição**”, “**Editora**”, “**Autor**” e “**Adquirido em**” são campos de texto estático
- A tabela contém dois Retângulos “arredondados enquanto os outros elementos são todos campos (fields)

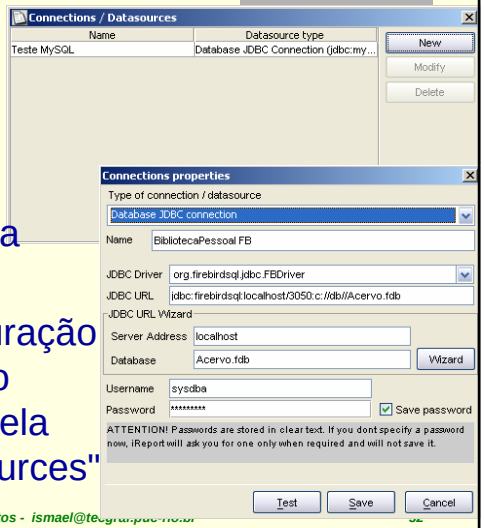


Definição da fonte de dados

- Para a criação do relatório, é preciso obter acesso aos metadados de uma **fonte de dados**.
- Para isso, é necessário definir essa fonte.
  - Os metadados da fonte são usados na definição do layout do relatório.
- O iReport provê suporte a diversas fontes de dados: JDBC, Hibernate, etc.

Fonte de dados via JDBC

- Configure a conexão
- testar se a conexão foi configurada corretamente.
- Clique em “Save” para salvar a conexão.
- Para alterar a configuração da conexão, clique no botão "Modify" da janela "Connections/Datasources"



Definir a consulta do relatório

- Após a definição da fonte de dados, devemos definir a **consulta do relatório**.
- Essa consulta tem os seguintes objetivos:
  - apresentar ao iReport os metadados;
  - permitir testar o relatório com informações provenientes da fonte de dados.
- Para definir a consulta, obtenha acesso à opção “Dados | Consulta do Relatório” na barra de menus do iReport.

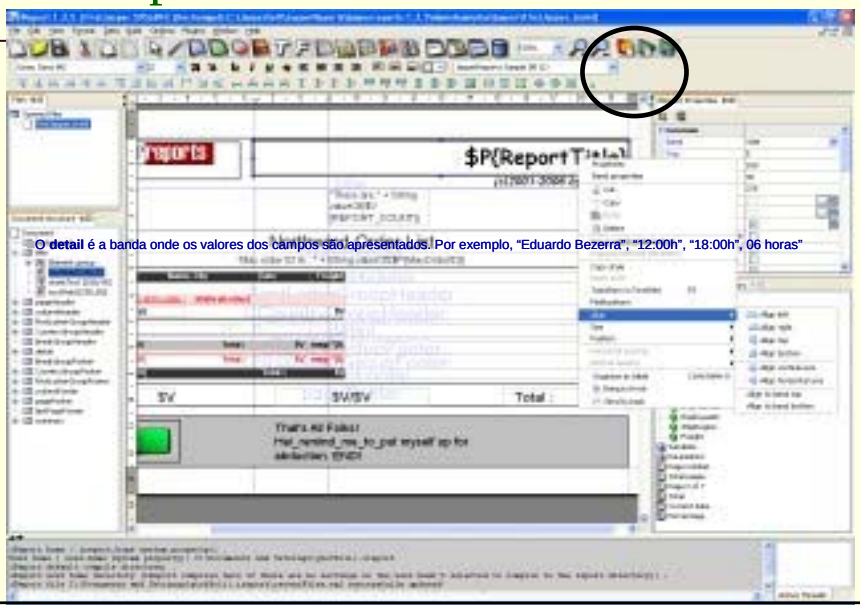
Desenhar o relatório

- Após a definição da consulta, os campos do resultado da consulta ficam disponíveis para desenhar o relatório.
  - Veja a janela “Library”
  - Expanda a pasta “Campos”
  - Arraste os campos desejados para a seção adequada do relatório (normalmente essa seção é a “details”)

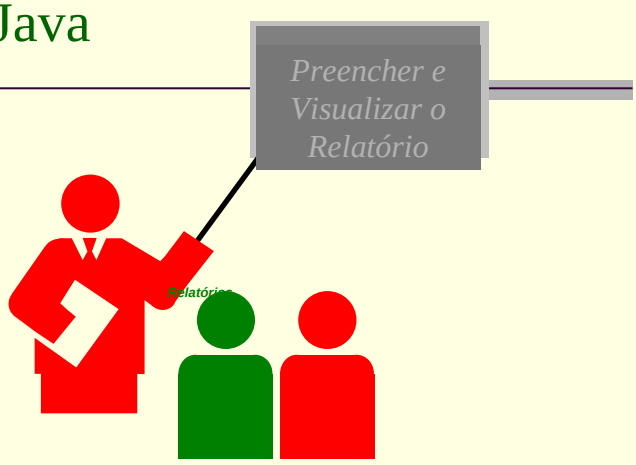
Compilar o relatório

- Após o desenho do relatório, devemos compilá-lo.
  - Esse processo corresponde a gerar uma representação em formato binário do relatório.
  - Resultado da compilação é um arquivo com a extensão .jasper.
  - É essa representação que usamos com o JasperReport para “preencher” o relatório.
- Para compilar o relatório, use a opção “Criar | Compilar” na barra de menus do iReport.
  - Há também um ícone na barra de ícones.

Compilar o relatório



FPSW-Java



Preencher o relatório

- Classe JasperFillManager
- Principal método: fillReport (estático)
  - Retorna objeto JasperPrint
- Assinatura:

```
JasperPrint fillReport(
    String sourceFileName,
    HashMap parameters,
    Connection connection);
```

Preencher o relatório (cont.)

- Parâmetros de fillReport:
  - String **sourceFileName**: corresponde ao nome do relatório compilado (.jasper).
  - HashMap **parameters**: lista de parâmetros do relatório (e.g., título, data, sessão)
  - Connection **connection**: conexão (JDBC) com a fonte de dados utilizada para preencher o relatório.

Exportar o relatório

- JasperExportManager
  - Classe útil quando queremos exportar o relatório para diversos formatos.
- Exemplos:

```
JasperExportManager.exportReportToHtmlFile(
    print, "hello.html");
JasperExportManager.exportReportToPdfFile(
    print, "hello.pdf");
JasperExportManager.exportReportToXmlFile(
    print, "hello.xml", false);
```

Visualizar o relatório

- JasperViewer
  - Classe útil para visualização do relatório.
  - O visualizador utilizado é do próprio Jasper
- O método **viewReport** (estático) é usado para apresentar o relatório no visualizador.
- Assinatura:

```
JasperViewer.viewReport(
        JasperPrint jasperPrint,
        boolean isExitOnClose)
```


Recursos relevantes

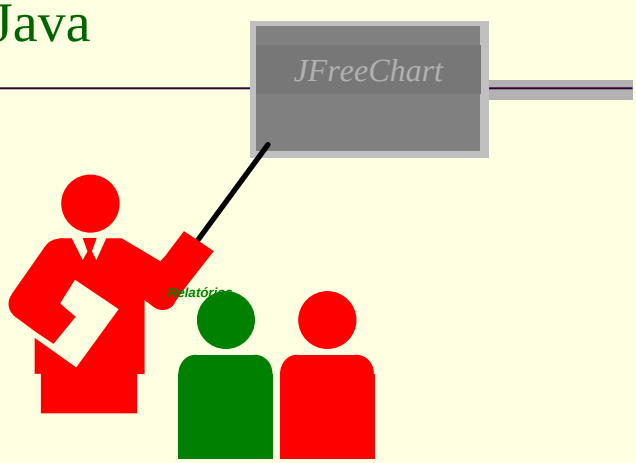
http://jasperreports.sourceforge.net/

http://jasperforge.org/sf/projects/ireport

http://www.jfreechart.org/

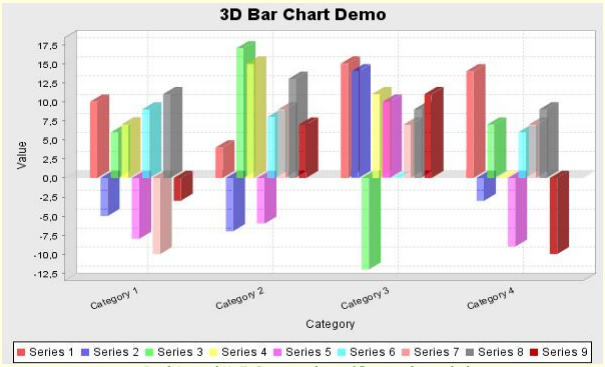
http://www.javafree.org/javabb/viewtopic.jbb?t=3154

FPSW-Java



O que é JFreeChart?

- JFreeChart é uma biblioteca livre para a Java utilizada em Aplicações Desktop, Applets, Servlets e JSP. http://www.jfree.org/jfreechart/index.html

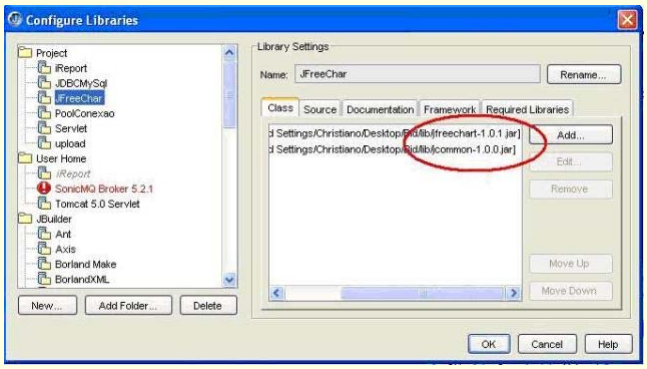


O que é JFreeChart?

- JFreechart pode ser usado para gerar gráficos de Pizza, gráficos de Barra, gráficos de linha (com ou sem efeito 3D), gráficos combinados, dentre diversos outros tipos de gráficos. Exporta dados para o formato PNG ou JPEG, Exporta para qualquer formato usando a implementação de Graphics2D incluindo:
  - PDF via iText (http://www.lowagie.com/iText/);
  - SVG via Batik (http://xml.apache.org/batik/);

Usando JFreeChart

- Os .jar's do JFreeChart são jcommon-1.0.12.jar e jfreechart-1.0.9.jar devem ser adicionados ao lib do projeto.

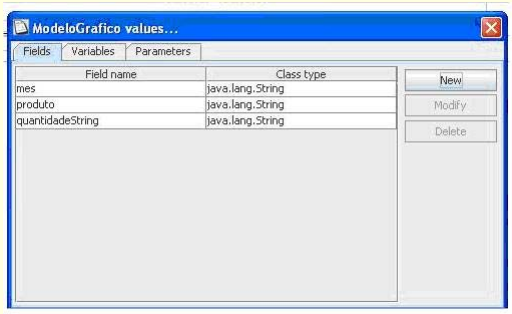


JFreeChart -> IReport

- Para anexar os gráficos do JFreeChart ao IReport é preciso instalar o plugin-netbeans do IReport. 1209691533171__iReport-nb-0.9.1.2.nbm
- **Gerando os Gráficos**
  - Primeiramente, vamos criar as classes ModeloGraficoItem que armazenará os dados que darão origem ao gráfico e ComposicaoDadosItemGrafico que gerará valores arbitrários que serão exibidos no gráfico:
    - *ModeloGraficoItem.java*
    - *ComposicaoDadosItemGrafico.java*

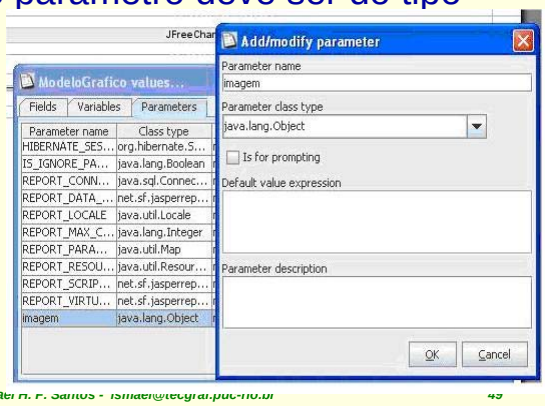
Criando modelo no IReport

- No IReport criamos um formulario para receber:
  - Os dados que geraram o gráfico apresentados em forma de listagem;
  - A imagem do gráfico;
- O relatório contém 3 fields para receber os valores que darão origem ao gráfico.



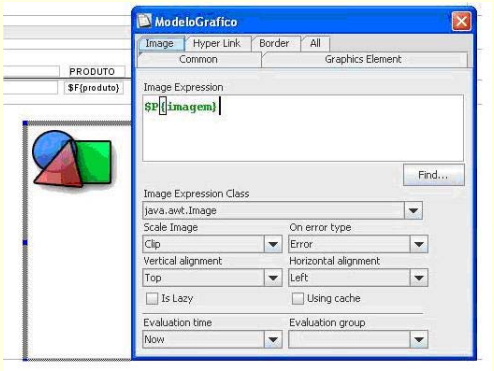
Modelo no IReport -> JFreeChart

- Nesta mesma tela, na aba Parameters vamos criar o parâmetro que irá receber a imagem. Observe que este parâmetro deve ser do tipo java.lang.Object:



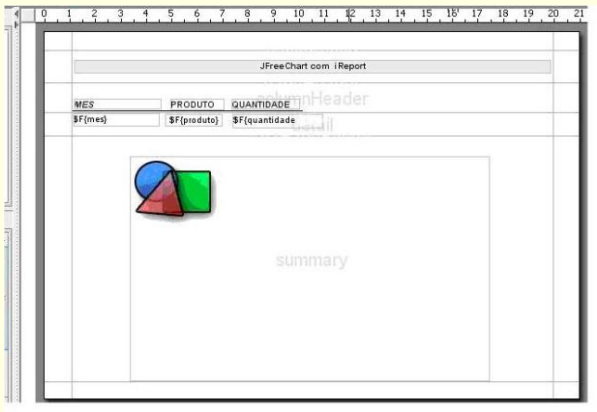
Modelo no IReport -> JFreeChart

- Para mostrar a imagem do gráfico vamos criar um campo na banda Sumary com a ferramenta ImageTools que receberá um objeto *imagem* por parâmetro:



Modelo no IReport -> JFreeChart

- A montagem final do Modelo ficou assim (salvo com o nome de ModeloGrafico): .



Modelo no IReport -> JFreeChart

- A aplicação devera ter uma classe cujas funcoes são:
  - Receber os valores gerados pela classe ComposicaoDadosItemGrafico;
  - Chamar a classe GeradorGrafico para criar um gráfico a partir dos da dados recebidos;
  - Chamar o *ModeloGrafico.jasper* exibindo o relatório em formato PDF em uma janela do browser.
- Para criar a imagem do gráfico vamos usar a classe BufferedImage que será o objeto adicionado ao parâmetro.

```
BufferedImage imagem =
    GeradorGrafico.gerarGraficoLinha3D("Titulo", "Mes",
    "Quantidade", array);
```

- A classe HashMap de java.util é usada para passar os parâmetros para o relatório:

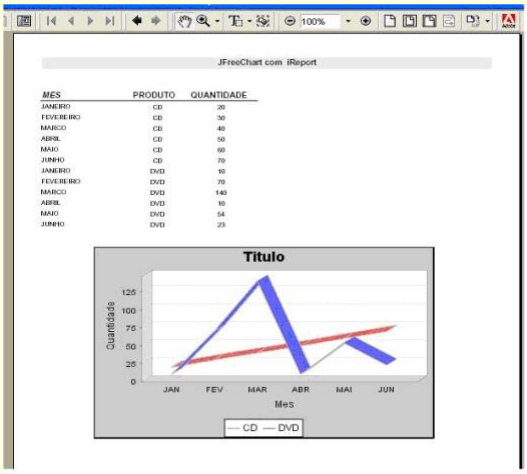
```
HashMap params = new HashMap();
params.put("imagem", imagem);
```

Modelo no IReport -> JFreeChart

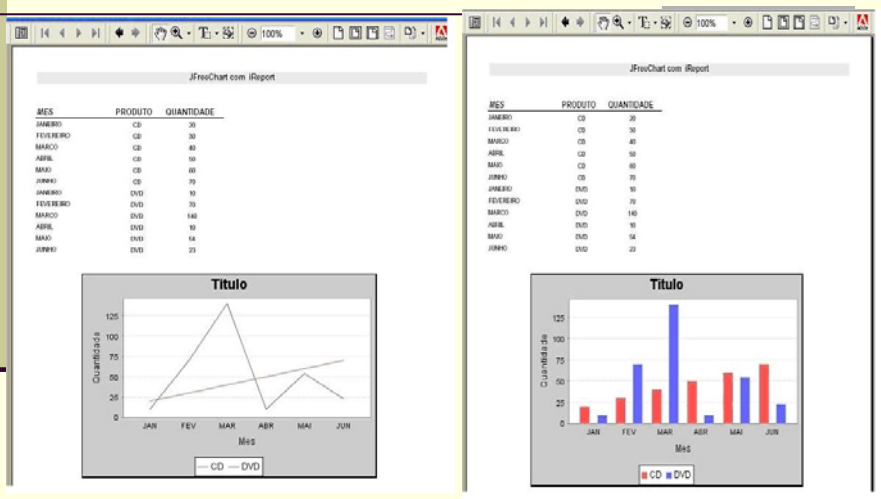
```
// A String pathJasper garda o caminho para o relatório
   compilado ModeloGrafico.jasper localizado na
   subpasta
//relatorio/grafico
   String pathJasper = "/TesteGrafico.jasper";
   JRDataSource jrDataSourse = new
   JRBeanArrayDataSource(array.toArray());
   Jasper Print impressao =
   JasperFillManager.fillReport(pathJasper, params,
   jrDataSourse);
  JasperExportManager.exportReportToPdf(impressao,
   "ModeloGrafico.pdf"););
```


Modelo no IReport -> JFreeChart

- A montagem final do Modelo ficou assim (salvo com o nome de ModeloGrafico):
  .

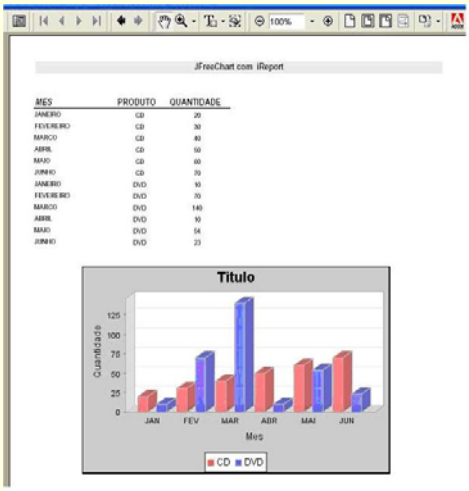


Modelo no IReport -> JFreeChart



Modelo no IReport -> JFreeChart

- JFreeChart pode ser usada juntamente com iReport / JasperReports criando relatórios mais completos usando também recursos de criação de gráficos.



Fonte de dados via Hibernate

- Configure o classpath (Options->classpath). Em particular, todos os itens a seguir devem estar disponíveis ao iReport:
  - arquivos de mapeamento (.hbm.xml)
  - arquivos das classes persistentes (.class)
  - arquivo de configuração do Hibernate
  - driver JDBC do SGBD a que ser quer ter acesso
- Após a configuração do classpath, configure a conexão (análoga à configuração via JDBC)