1. **INTRODUÇÃO**

O Jasper é a principal biblioteca Java utilizada na Vixteam para a criação de relatórios em PDF, ele também permite a criação de relatórios .xls mas em algumas situações para este tipo de relatórios o Apache Poi é utilizado não sendo o uso do Jasper priorizado como no caso dos arquivos PDF.

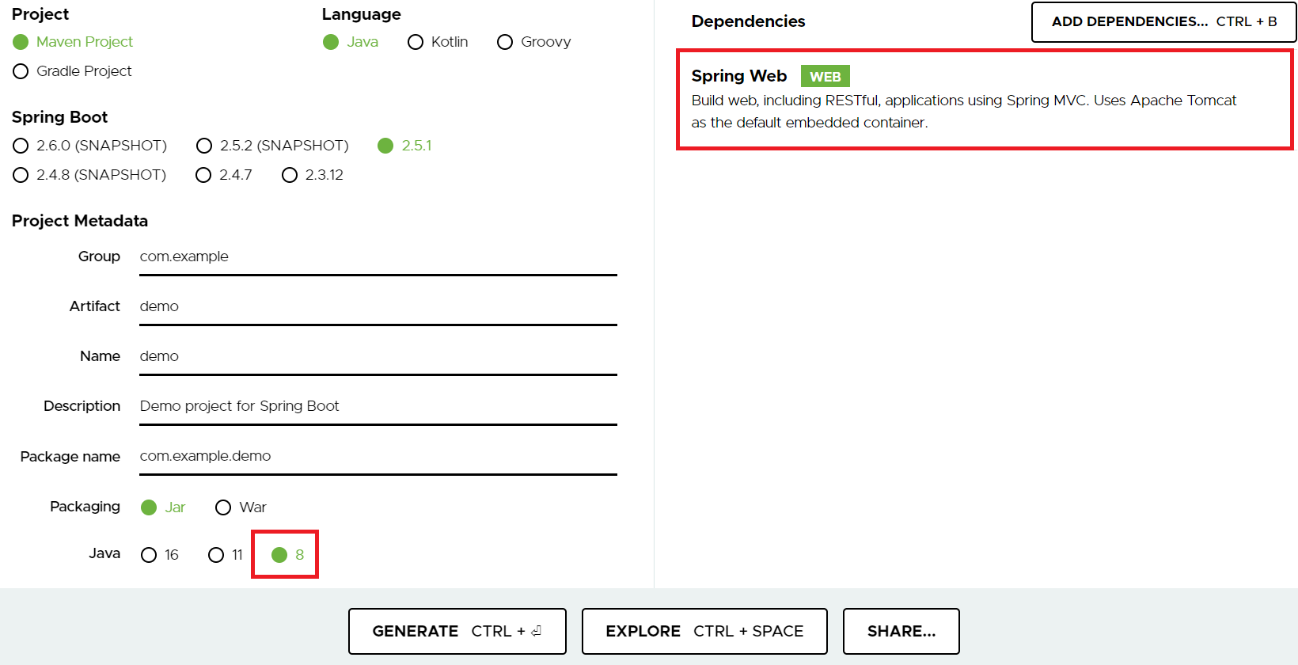
Dentre as características principais do Jasper para criação de relatórios podemos destacar:

* Existência de um editor drag & drop para a criação dos relatórios
* Possibilidade de utilizar o mesmo relatório para diferentes formatos ex (PDF e EXCEL)]
* Flexibilidade quando a origem dos dados do relatório que pode ser obtida por via de parâmetros passados pelo Java ou a partir de uma query embutida dentro do próprio relatório

1. **CRIANDO PROJETO JAVA COM SPRING BOOT**

Para nossa maior praticidade deste treinamento iremos criar um projeto do zero utilizando um inicializador web para projetos SpringBoot e em cima deste projeto aplicaremos nosso treinamento e nossos exemplos de utilização do Jasper.

Logo acesse o site <https://start.spring.io/> e crie um projeto conforme a imagem abaixo, criado o projeto abra-o na sua IDE de preferência (No nosso material iremos abordar o uso da IDE InteliJIdea). Obs: Lembre-se de abrir o projeto pelo seu pom.xml abrindo-o como projeto Maven.



1. **INSTALAÇÃO DO JASPER**

Para incluir a biblioteca do Jasper em seu projeto recomendamos que você utilize o Maven e com ele adicione a dependência da biblioteca, por exemplo, para adicionar o jasper 6.17.0 basta incluir a dependência abaixo:

<**dependency**>  
 <**groupId**>net.sf.jasperreports</**groupId**>  
 <**artifactId**>jasperreports</**artifactId**>  
 <**version**>6.17.0</**version**>  
</**dependency**>

Instalado o .jar da biblioteca em seu projeto o próximo passo é baixar o editor de relatórios do Jasper, o download desta ferramenta pode ser feito em <https://community.jaspersoft.com/project/jaspersoft-studio/releases>

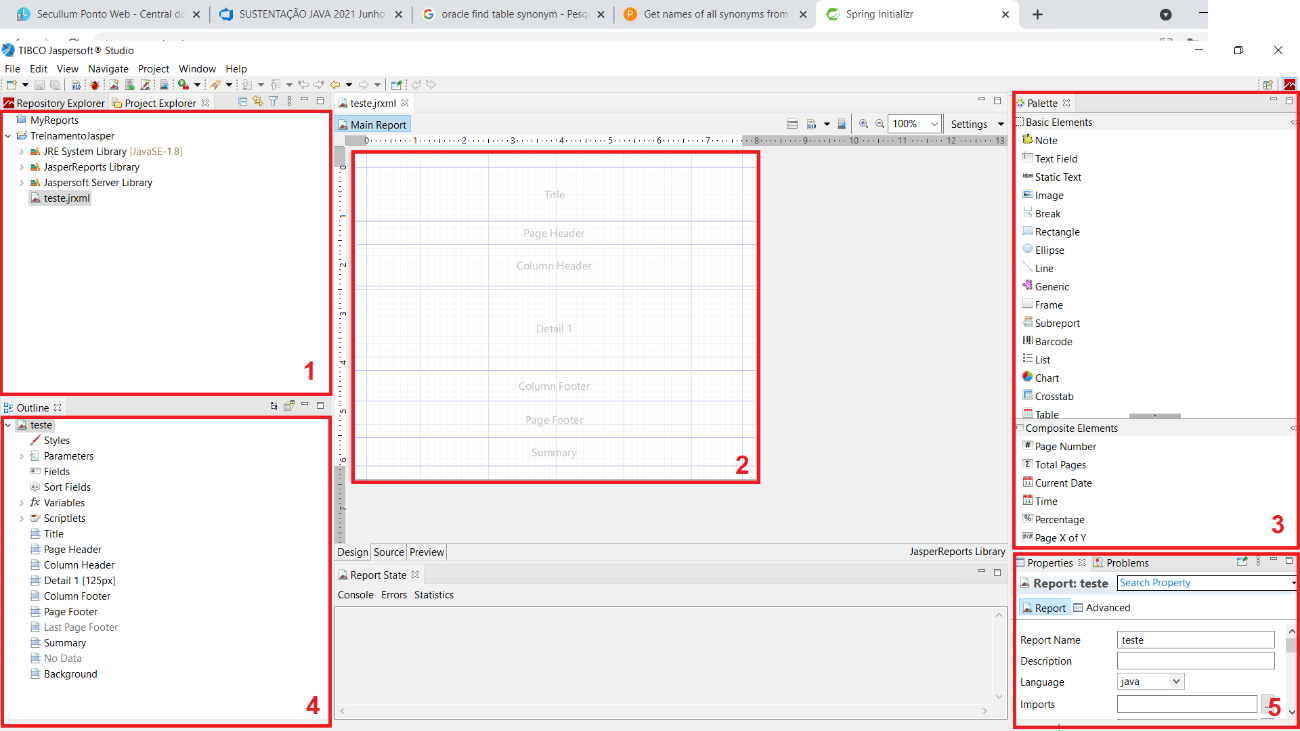
Ao realizar o donwload do edior de relatórios Jasper Studio **é extremamente importante** que a versão do editor seja igual à versão do jar da biblioteca do Jasper, caso contrário erros de compatibilidade poderão ocorrer quando o relatório for executado em sua aplicação.

**3.0 EDITOR DE RELATÓRIOS JASPER**

Através do Jasper Studio você poderá arrastar diferentes componentes para diferentes regiões de seu relatório e assim construí-lo de forma que posteriormente em sua aplicação Java você irá referenciá-lo para assim entregar ao usuário final o relatório construído no formato selecionado em seu código.

A ferramenta é bem intuitiva e por esse motivo não iremos entrar em grandes detalhes sobre como utilizá-la neste guia.

Para começar a trabalhar crie um Projeto Jasper e nele crie um arquivo Jasper Report, abrindo-o logo em seguida. Você irá visualizar uma tela como a da imagem abaixo:



1. Nesta região você poderá navegar no seu Projeto de relatórios Jasper
2. Aqui é uma representação do seu relatório, arraste elementos para esta região e construa seu relatório de forma dinâmica
3. Aqui são diversos componentes que você poderá utilizar em seu relatório, basta clicar, segurar e arrasta-los até a região 2.
4. Aqui temos diversos campos e parâmetros que podem ser utilizados em seus relatórios, os campos desta região vem da consulta do relatório que pode ser embutida no mesmo ou que pode ter vindo de uma lista a partir do seu projeto Java.
5. Nesta região irá aparecer as propriedades do elemento atualmente selecionado.

No Jasper Studio, outro ponto importante que você precisa ter em mente é que seu relatório (Item 2) é dividido em algumas regiões que possuem comportamentos especiais, destacando-se:

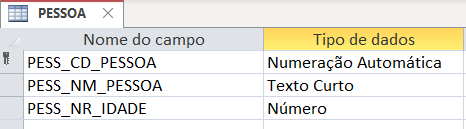
* **Title:** Será a primeira região a aparecer no seu relatório, esta região aparecerá uma única vez.
* **Page Header:** Aparece no topo de cada página do seu relatório.
* **Column Header:** Aparece uma única vez ao topo da região de Detail associada
* **Detail:** Esta região é repetida uma vez para cada ocorrência de registro da consulta do relatório
* **Column Footer:** Aparece uma única vez ao fim da região de Detail associada
* **Page Footer:** Aparece na parte inferior de cada página do seu relatório.
* **Summary:** Será a última região a aparecer no seu relatório, esta região aparecerá uma única vez.

1. **FONTE DE DADOS DO RELATÓRIO**

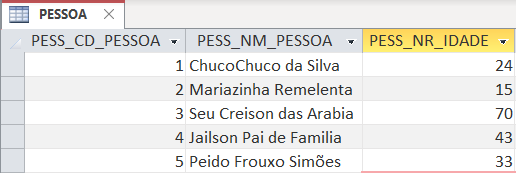
Associado a todo relatório Jasper existe uma consulta que pode estar embutida no relatório ou vir de uma aplicação Java.

Para vermos como este recurso funciona iremos utilizar o Access para ser nosso “Banco de dados”, logo crie um arquivo Access onde nele exista a tabela PESSOA com os campos PESS\_CD\_PESSOA, PESS\_NM\_PESSOA, PESS\_NR\_IDADE. Aproveite e na tabela PESSOA insira pelo menos 5 dados diferentes, veja:

**Estrutura:**



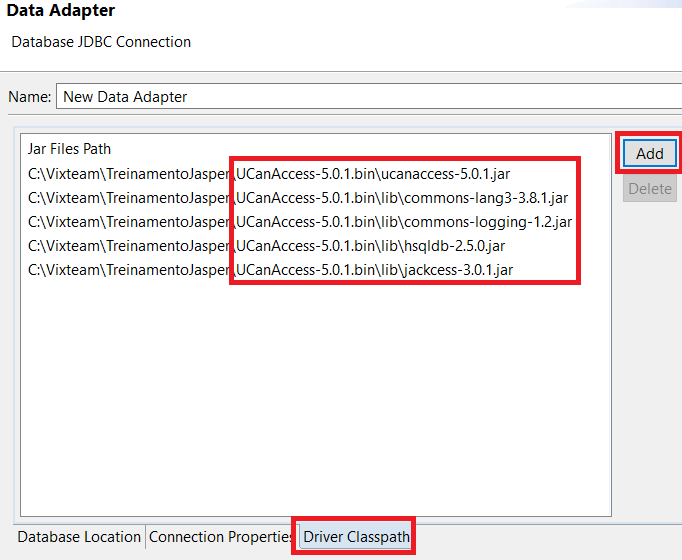
**Dados:**



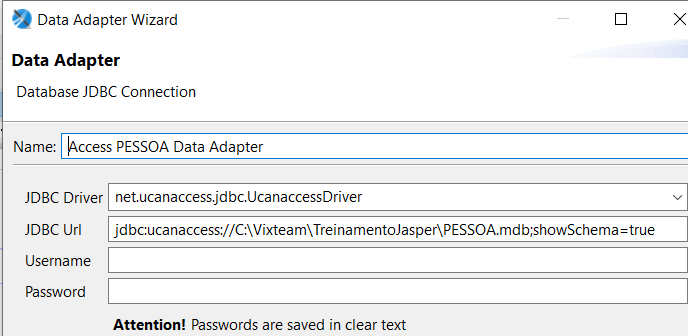
**4.1 CONSULTA EMBUTIDA NO PRÓPRIO RELATÓRIO**

Primeiro precisamos conectar nossa “base de dados” Acces com o relatório, para isso siga os passos:

1. Na aba Repository Explorer, clique com botão direito nos Data Adapters e selecione Create Data Adapter.
2. Selecione Database JDBC Connection.
3. Informe um nome para o driver.
4. Na guia Driver Classpath, clique em Add. E adicione o .Jar com o driver de conexão a banco de dados Access para aplicações Java. Iremos utilizar o JDBC Access UCanAccess que pode ser obtido em <http://ucanaccess.sourceforge.net/site.html> baixado o arquivo, você terá que adicionar os seguintes .jars:

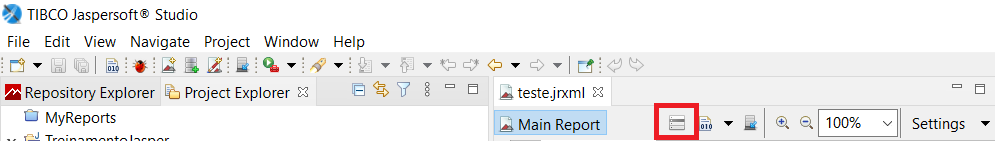


1. Agora na guia Database Location, entre o nome da classe do driver "net.ucanaccess.jdbc.UcanaccessDriver”
2. Informe a JDBC URL com o caminho do seu arquivo, Ex: jdbc:ucanaccess://C:\Vixteam\TreinamentoJasper\PESSOA.mdb;showSchema=true

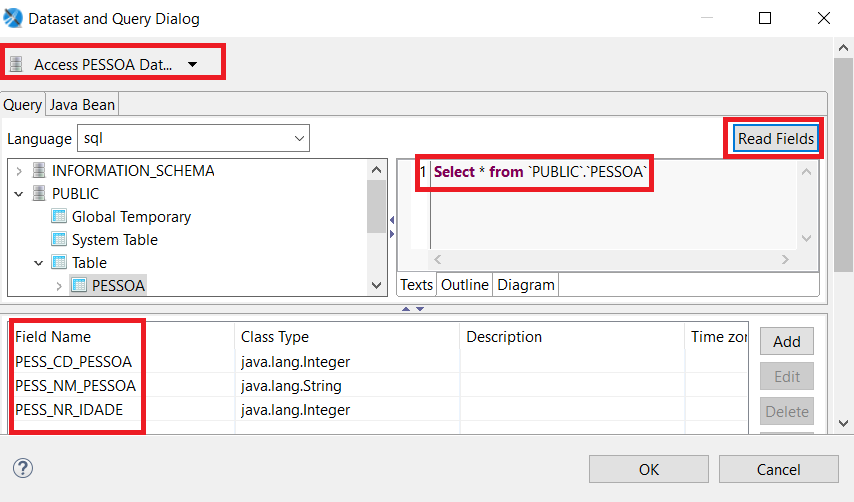


1. Agora teste a conexão para ver se tudo ocorreu corretamente

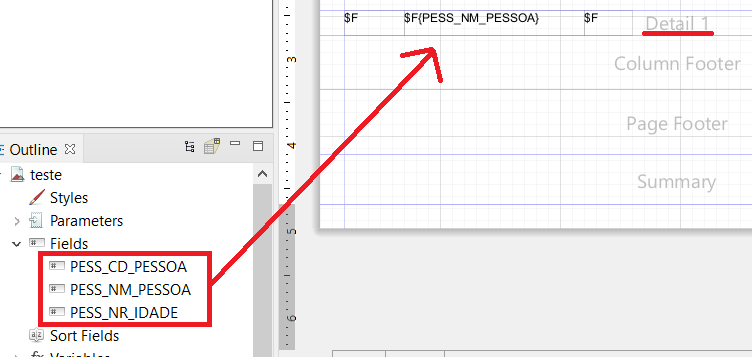
Finalizada a configuração do adaptador de dados, acesse o editor de querys do jasper conforme imagem abaixo:

****

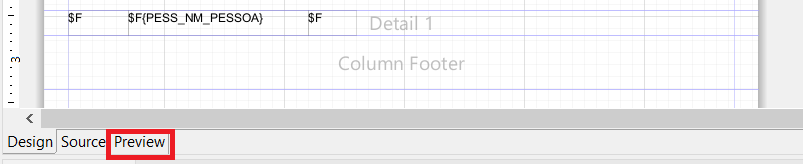
Na janela aberta, conecte ao adapter do access, informe uma query e clique sobre “Read Fields”, desta forma, cada um dos campos obtidos de sua query será um campo que você poderá utilizar na construção do seu relatório, veja:

****

Com isso, e arrastando os 3 campos descobertos para dentro da região Detail de seu relatório, perceba que você conseguirá gerar um relatório que exibe o conteúdo que você definiu em sua tabela PESSOA no Access.



Arrastado os campos, para gerar seu relatório clique na aba Preview conforme imagem abaixo:

****

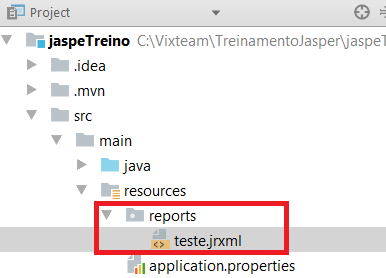
**4.2 CONSULTA VINDO DE UMA APLICAÇÃO JAVA**

Outra maneira que temos de utilizar o Jasper é realizando a consulta na aplicação Java e mandando através de uma lista o resultado da consulta para o Jasper, antes de chegarmos neste ponto veremos primeiro no próximo capítulo como fazemos para utilizar nosso relatório Jasper em uma aplicação Java.

**4.2.1 RELATORIO JASPER COM CONSULTA EMBUTIDA EM APLICAÇÃO JAVA**

Para a partir de uma aplicação Java conseguirmos utilizar um relatório Jasper será necessário a execução dos passos a seguir:

* Adicionar a dependência do Jasper conforme descrito no capítulo 3.0
* No projeto gerado pelo capítulo 2.0, adicione a dependência do JDBC para conexão ao banco Access:
  + <**dependency**>  
     <**groupId**>net.sf.ucanaccess</**groupId**>  
     <**artifactId**>ucanaccess</**artifactId**>  
     <**version**>5.0.1</**version**>  
    </**dependency**>
* Copie o arquivo .jrxml do relatório a ser adicionado na aplicação Java e mova-o para o diretório:



* Agora para obter a conexão com o banco Access crie o seguinte método:
  + **private** Connection getAccessConnection(){  
     **try**{  
     String database=**"C://Vixteam//TreinamentoJasper//PESSOA.mdb"**;*//Here database exists in the current directory* String url=**"jdbc:ucanaccess://"** + database + **";showSchema=true"**;  
     **return** DriverManager.*getConnection*(url);  
     }**catch**(Exception ee){System.***err***.println(ee);}  
     **return null**;  
    }
* Agora criaremos um método para obter um objeto do tipo JasperPrint que é uma representação do relatório Jasper posteriormente utilizada para a geração de seu PDF. Perceba que passamos para o relatório apenas a conexão com nosso banco Access, sem necessidade de elaborar uma query SQL, isto ocorre pois nossa query está embutida no relatório conforme fizemos no capítulo 4.1:
  + **private** JasperPrint exportPdfFile() **throws** Exception {  
     String path = **resourceLoader**.getResource(**"classpath:reports\\teste.jrxml"**).getURI().getPath();  
     JasperReport jasperReport = JasperCompileManager.*compileReport*(path);  
      
     *// Parameters for report* Map<String, Object> parameters = **new** HashedMap();  
      
     **return** JasperFillManager.*fillReport*(jasperReport, parameters, **this**.getAccessConnection());  
    }
* Por fim utilizamos o Jasper e o Spring para mapearmos uma requisição e respondermos à mesma com um arquivo PDF gerado baseando-se no relatório que criamos anteriormente:
  + @RequestMapping(value=**"/testePdf"**)  
    **public void** testePdf(ModelAndView model, HttpServletResponse response) **throws** Exception {  
     response.setContentType(**"application/x-download"**);  
     response.setHeader(**"Content-Disposition"**, String.*format*(**"attachment; filename=\"testePdf.pdf\""**));  
      
     JasperPrint jasperPrint = **this**.exportPdfFile();  
     OutputStream out = response.getOutputStream();  
     JasperExportManager.*exportReportToPdfStream*(jasperPrint, out);  
    }
* Para testar esta implementação, inicie sua aplicação e no navegador chame por http://localhost:8080/testePdf

**4.2.2 RELATORIO JASPER COM CONSULTA NA APLICAÇÃO JAVA**

Veremos agora neste capítulo como implementamos um relatório cujos dados de sua consulta serão obtidos a partir de uma aplicação Java. Para começarmos crie uma classe Pessoa que irá servir para representar um objeto PESSOA do nosso banco Access, é importante que a classe possua os métodos get e set para cada uma de suas propriedades:

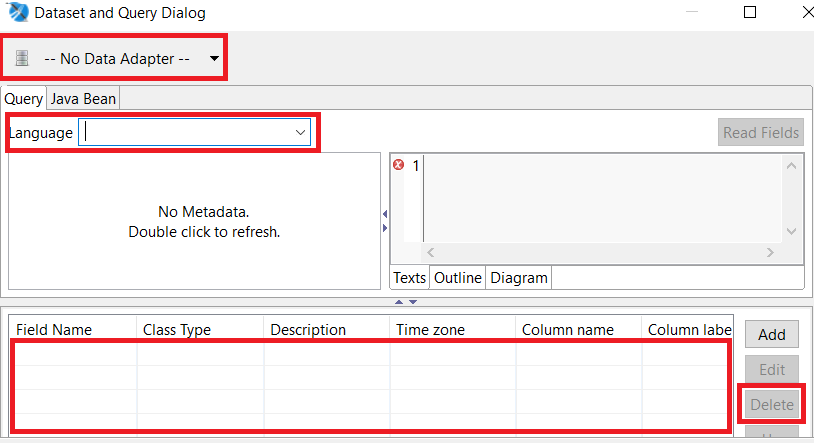
**public class** Pessoa {  
 **private** Long **codigo**;  
 **private** String **nome**;  
 **private** Integer **idade**;  
  
 **public** Long getCodigo() {  
 **return codigo**;  
 }  
  
 **public void** setCodigo(Long codigo) {  
 **this**.**codigo** = codigo;  
 }  
  
 **public** String getNome() {  
 **return nome**;  
 }  
  
 **public void** setNome(String nome) {  
 **this**.**nome** = nome;  
 }  
  
 **public** Integer getIdade() {  
 **return idade**;  
 }  
  
 **public void** setIdade(Integer idade) {  
 **this**.**idade** = idade;  
 }  
}

Agora, no Jasper Studio, acesse o editor de Dataset e Querys do Jasper conforme imagem abaixo:

Interface gráfica do usuário, Aplicativo, Word

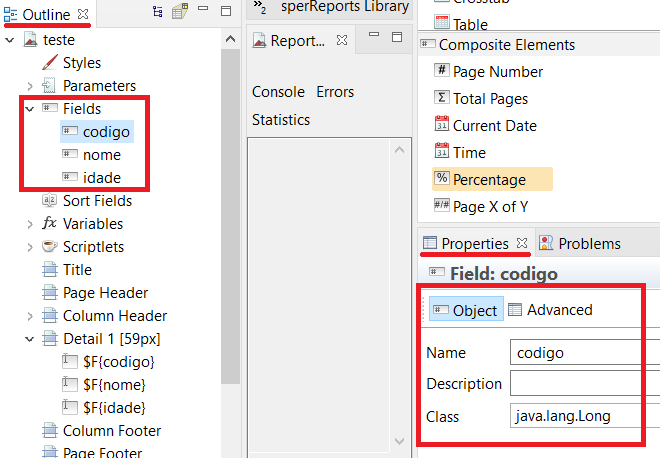
Descrição gerada automaticamente

Como copiamos nosso relatório, remova todos os campos que já existirem e remova também o Data Adapter:



Devemos agora manualmente adicionar cada um dos campos do objeto que será enviado na listagem para o Jasper, em nosso exemplo iremos mandar um List<Pessoas> desta forma devemos manualmente criar os campos codigo, nome e idade (Utilizando os mesmos nomes das propriedades no objeto Java).

Para criar um campo, no JasperStudio, na janela de outline, clique com o botão direito em Fields e depois Create Field, posteriormente ajustando o nome e tipo do campo criado na parte de properties para que ele fique coerente com a propriedade do objeto java Pessoa:



Agora salve este novo relatório e copie-o para o seu projeto.

Nos fontes Java, altere o método exportPdfFile que criamos anteriormente conforme a seguir:

**private** JasperPrint exportPdfFile() **throws** Exception {  
 String path = **resourceLoader**.getResource(**"classpath:reports\\teste-out.jrxml"**).getURI().getPath();  
 JasperReport jasperReport = JasperCompileManager.*compileReport*(path);  
  
 *// Parameters for report* Map<String, Object> parameters = **new** HashedMap();  
  
 List<Pessoa> lst = **new** ArrayList<>();  
 lst.add(**new** Pessoa(1l, **"Joao"**, 20));  
 lst.add(**new** Pessoa(2l, **"Maria"**, 33));  
 JRBeanCollectionDataSource beanColDataSource = **new** JRBeanCollectionDataSource(lst);  
  
 **return** JasperFillManager.*fillReport*(jasperReport, parameters, beanColDataSource);  
}

Veja que com esta alteração, ao invés de passarmos a conexão Access anteriormente utilizada para executar a consulta embutida do nosso relatório passamos agora um List<Pessoa> dentro do objeto JRBeanCollectionDataSource, desta forma, os dados do relatório serão obtidos desta listagem que por sua vez em uma aplicação java pode ser obtida a partir de uma consulta SQL da aplicação.