

Utilizando os Relatórios do pyHed.

Nesta documentação vamos aprender a construir um relatório utilizando o `frameCustomRptCondition`. Este frame foi desenvolvido para que o programador desenvolva relatórios com facilidade, rapidez e confiabilidade. Neste documento não detalharemos o procedimento de criação de uma tela (Criação do arquivo => Inserção no `frameMain` => Inserção da ação), consideramos que se você está construindo um relatório, já construiu uma tela de cadastro ou gerência onde no tutorial das mesmas é detalhado.

Então vamos a construção do nosso arquivo. Existem dois métodos que são vitais na construção do relatório. São eles: `__init__` e `evtBeforeBuild`. Agora, vamos detalha-los:

1º Passo - `__init__()`: É neste método que vamos configurar `MasterFields`, `DatailFields` e a SQL do relatório. Veja o exemplo de código abaixo:

```
def __init__(self, parent, params=None):

    # Nome que será salvo o relatório.
    self.reportPath = appConsts.pdfPath + '/' + str(time.strftime("%Y%m%d%H%M%S")) + '.pdf'

    # Campos para exportação de csv.
    exportFields = [ ['*ID_PESSOA', 'Id Pessoa'],
                    ['COD_PESSOA', u'Cód Pessoa'],
                    ['NOME', 'Nome'], ['SEXO', 'Sexo'],
                    ['DATA_NASCIMENTO', 'Data Nascimento'],
                    ['OBSERVACAO', u'Observação'],
                    ['TIME_FUTEBOL', 'Time Futebol'] ]

    # Lista/Tupla dos Campos, a Ordenação de Exibição parte daqui, setando campo e o
    # label que será exibido no relatório.
    listMasterFields = list()
    listMasterFields.append({"*ID_PESSOA" : "Id Pessoa:"})
    listMasterFields.append({"NOME" : "Nome:"})
    listMasterFields.append({"SEXO" : "Sexo:"})
    listMasterFields.append({"DATA_NASCIMENTO" : "Data Nascimento:"})
    listMasterFields.append({"OBSERVACAO" : "Observação:"})

    listDetailFields = list()
    listDetailFields.append({"COD_PESSOA" : "Cód Pessoa"})
    listDetailFields.append({"TIME_FUTEBOL" : "Time Futebol"})

    # Define os Master e Detail Fields do Relatório.
    self.reportFields = {"masterFields" : listMasterFields, "detailFields" : listDetailFields}

    super(FrameRelatorioPessoas, self).__init__(parent,
                                                reportPath=self.reportPath,
                                                reportFields=self.reportFields,
                                                exportCsvFields=exportFields,
                                                params=params)
```

```

self.SQL = """
select
    P.ID_PESSOA AS ID_PESSOA,
    P.COD_PESSOA AS COD_PESSOA,
    P.NOME AS NOME,
    case
        when upper(P.SEXO) = 'M' then 'Masculino'
        when upper(P.SEXO) = 'F' then 'Feminino'
        else 'Não Informado'
    end SEXO,
    ( select
        case
            when FORMAT_DATE is not null then FORMAT_DATE
            else 'Não Informado.'
        end FORMAT_DATE
    from
        DATE_TO_CHAR(P.DATA_NASCIMENTO)
    ) as DATA_NASCIMENTO,
    P.OBSERVACAO AS OBSERVACAO,
    CF.NOME AS TIME_FUTEBOL
from
    PESSOA P inner join
    CLUBE_FUTEBOL CF on P.ID_TIME_FUTEBOL = CF.ID_CLUBE
where
    %WHERECLAUSE%"""

self.title = u'Relatório de Pessoas'

```

Agora vamos conhecer parâmetro a parâmetro:

- **self.reportPath:** Deve ser informado o caminho completo do arquivo incluindo o nome. Ex: C:\reports\myReport.pdf;
- **exportsFields:** Lista com tuplas seguindo este padrão ['CAMPO', 'LABEL'], onde:
 - CAMPO: É o nome do campo na select;
 - LABEL: Label que irá aparecer no relatório.
- **listMasterFields:** Lista/Tupla com a ordem dos MasterFields do relatório;
- **listDetailFields:** Lista/Tupla com a ordem dos DetailFields do relatório;
- **self.reportFields:** Define os Master e Detail Fields do Relatório. Note que é uma lista com duas tuplas, o primeiro valor de cada tupla não pode ser alterado, sendo somente informado o segundo valor. O segundo é uma lista, neste caso as citadas acima.

Feito isto, podemos executar o `__init__` da herança passando os parâmetros necessários. Veja o trecho abaixo:

```

super(FrameRelatorioPessoas, self).__init__(parent,
reportPath=self.reportPath,
reportFields=self.reportFields,
exportCsvFields=exportFields,
params=params)

```

Após o `__init__` da herança. São informados mais 2 parâmetros:

- **self.SQL:** Deve ser informado a SQL do relatório. Importante lembrar que SEMPRE deve haver a clausula `%WHERECLAUSE%` ela será substituída em tempo de execução pelas cláusulas criadas pelo usuário.
Obs.: No futuro o pyHed permitirá o programador informar a classe mapeada no ORM ao invés de uma SQL;
- **self.title:** Informar o título da aplicação.

2° Passo - `evtBeforeBuild(self, report)`: Este evento, como seu próprio nome diz, é executado antes da construção do relatório. Seu único parâmetro é a instância do relatório, a partir deste parâmetro temos acesso a todas propriedades da herança, tornando o framework de relatórios do pyHed mais flexível. Segue as propriedades:

- **MasterColumnsMemo:** Coluna(s) do(s) MasterField(s) cujo texto é um Memo;
- **Encoding:** Encoding do relatório, usado para conversão nos textos;
- **SavePath:** Local e nome a ser salvo o Arquivo PDF;
- **ImagePath:** Local e nome da imagem usada no cabeçalho;
- **QtdColumnsMaster:** Quantidade de itens no cabeçalho;
- **MasterKey:** Chave primaria do relatório, separa um registro do outro;
- **HeadersMaster:** Label dos MasterFields caso estejam Separados;
- **FieldsHeadersMaster:** Campos que pertencem a cada label do cabeçalho;
- **ReportHeaders:** Títulos como data, hora e página do cabeçalho, habilitando ou não cada item;
- **TotalizerLabels:** Texto a ser exibido no somatório da ocorrência de uma palavra nos masters;
- **TotalizerColumnSum:** Texto a ser exibido no somatório da ocorrência de uma palavra nos details;
- **TotalizerColumnCount:** Texto a ser exibido no somatório geral das colunas dos details;
- **DetailRegistersSum:** Soma colunas definidas exibindo por registro;
- **MasterColumnsCounter:** Soma a ocorrência de palavras nos MasterFields;
- **Title:** Título do relatório encontrado no cabeçalho;
- **Footer:** Rodapé do relatório;
- **DataSet:** Array contendo os registros da consulta;
- **DetailsCounters:** Campos para somatório de ocorrência de uma ou mais palavras nos details, exibindo no final do relatório;
- **DetailsSum:** Campos para somatório de valores das colunas dos details exibindo no final do relatório;
- **DetailsCodeBar:** Colunas details que é código de barras;
- **DetailColumnGroup:** Coluna dos details que agrupa por valor os registros;
- **ReportFields:** Property que define Master/Detail;
- **FlagHeader:** Flag que define a exibição do cabeçalho;
- **FlagFooter:** Flag que define a exibição do rodapé.
- **FlagCounter:** Flag que Define a Exibição dos Contadores e Somatórios.

Agora que conhecemos as propriedades vamos ver um exemplo de código.

```

def evtBeforeBuild(self, report):

    # Título do Relatório
    report.Title = "Relatório de Pessoas"
    # Label's do Cabeçalho do Relatório
    report.ReportHeaders = {"DATE" : "Data:", "PAGE" : "Página:", "HOUR" : "Hora:"}
    # Campos do Tipo Memo do Relatório
    report.MasterColumnsMemo = {"OBSERVACAO" : True}
    # Rodapé do Relatório
    report.Footer = "www.pyhed.com"

    # Lista Com o Label da Divisão dos Master Fields
    listHeaderMaster = list()
    listHeaderMaster.append({"PESSOA" : "Informações da Pessoa"})
    # Define a Tupla com Os Label's dos Master Fields
    report.HeadersMaster = listHeaderMaster

    # Lista com os Campos dos MasterFields que pertencem a cada Label
    listFieldsTarefa = list()
    listFieldsTarefa.append("ID_PESSOA")
    listFieldsTarefa.append("NOME")
    listFieldsTarefa.append("SEXO")
    listFieldsTarefa.append("DATA_NASCIMENTO")
    listFieldsTarefa.append("OBSERVACAO")

    # Define quais campos pertecem a cada label dos masterfields.
    report.FieldsHeadersMaster = {"PESSOA" : listFieldsTarefa}

    # Define uma Coluna dos Details ou Mais como Código de Barras.
    report.DetailsCodeBar = {"COD_PESSOA" : True}

    # Define a Coluna dos Details que deverá ser somada setando um valor inicial.
    report.DetailRegistersSum = {"COD_PESSOA" : 0}

    # Título do Somatório de Ocorrencias de Uma Palavra na Coluna dos MasterFields.
    report.TotalizerLabels = "Somatório de Master Fields"

    # Lista com os Textos a Serem Pesquisados no Relatório.
    listNomes = list()
    listNomes.append("João da Silva Gulart")

    # Define a Coluna em que deve pesquisar e quais textos deve pesquisar na coluna
    #gerando um somatório.
    report.MasterColumnsCounter = {"NOME" : listNomes}

    # Define o Título do Somatório de Ocorrência de uma Palavra nos DetailFields.
    report.TotalizerColumnCount = "Somatório de Ocorrências nos Detail Fields"

    # Lista com os texto a serem pesquisados na coluna.
    listTime = list()
    listTime.append("Grêmio")

    # Define a Coluna dos Details a ser pesquisada e seta os itens da pesquisa.
    report.DetailsCounters = {"TIME_FUTEBOL" : listTime}

    # Faz um agrupamento dos details por uma coluna dos details, agrupa pelos valore iguais.
    report.DetailColumnGroup = "TIME_FUTEBOL"

    # Somatório de um Campo Numérico nos details.
    report.TotalizerColumnSum = "Somatório de Colunas de Detail Fields"

    # Define qual coluna dos details somar setando um valor inicial.
    report.DetailsSum = {"COD_PESSOA" : 0}

```

3º Passo – onPaint(): É neste método que vamos declarar os componentes que farão o filtro na geração do relatório. Estes componentes são da biblioteca condComponents do pyHed, foram construídos exclusivamente para gerar filtros nos relatórios, apenas declarando-os o seu relatório fará os filtros automaticamente. Segue o exemplo de código.

```
def onPaint(self):
    super(FrameRelatorioPessoas, self).onPaint()

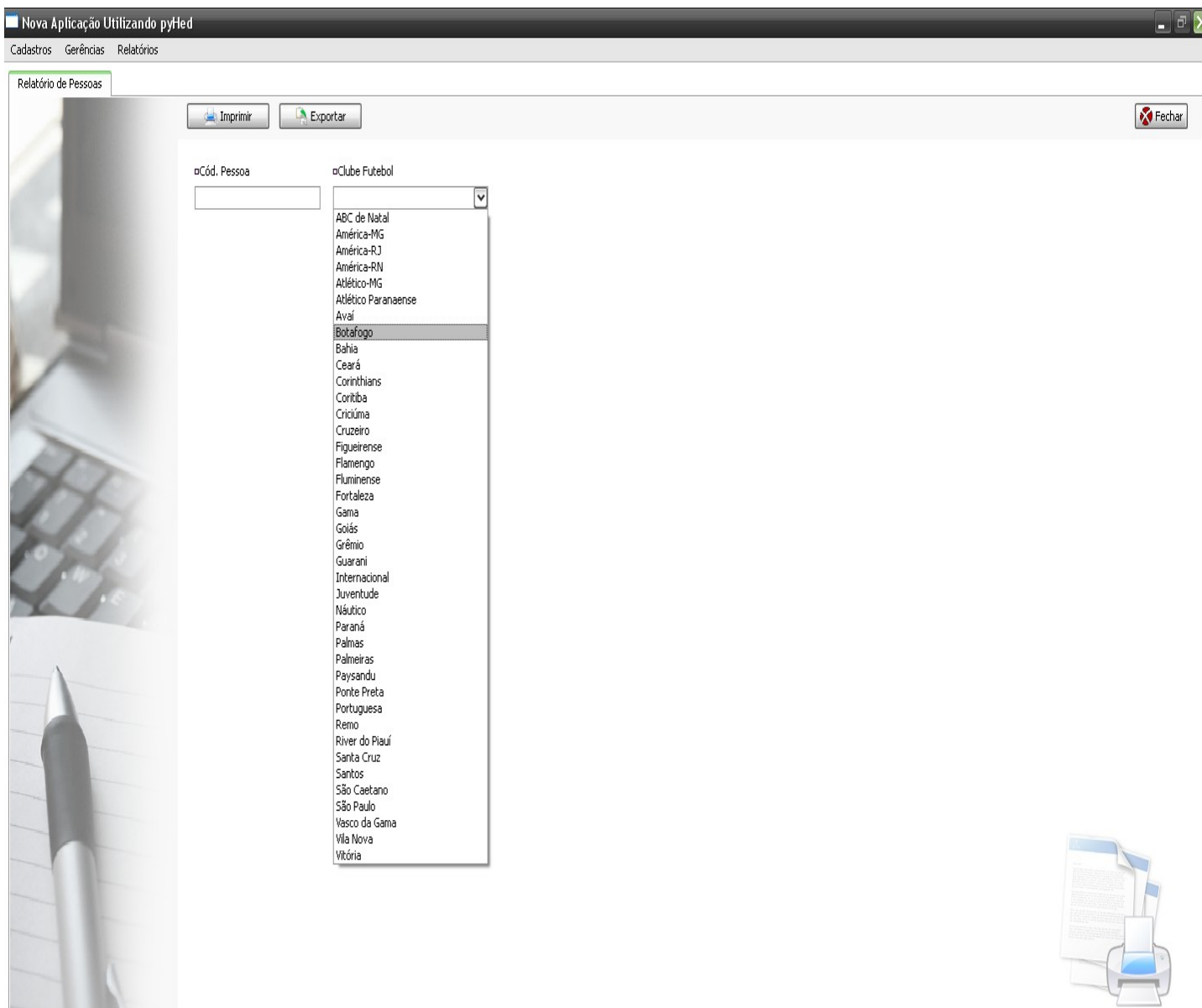
    self.edtNro = condComponents.CondNumericEdit(self.PnlMain,
                                                condition="P.COD_PESSOA = %VALUE%",
                                                required=False, text=u'&Cód. Pessoa',
                                                x=20,
                                                y=20,
                                                width=150)

    self.strSQL = 'select ID_CLUBE, NOME from CLUBE_FUTEBOL order by ID_CLUBE'
    items = pyHedConsts.dbInst.getConnection().execute(self.strSQL).fetchall()
    self.lkpClube = condComponents.CondDbLookupComboBox(self.PnlMain,
                                                        condition="CF.ID_CLUBE = %VALUE%",
                                                        rsItems=items,
                                                        text='&Clube Futebol',
                                                        x=(self.edtNro.x() + self.edtNro.width() + 15),
                                                        y=20,
                                                        width=185)

    self.lkpClube.setreadOnly(False)
```

Note que a única diferença destes componentes é o parâmetro " condition ", neste parâmetro informe o filtro que deve ser feito na SQL, NUNCA esquecendo do %VALUE%, ele que será substituído em tempo de execução.

Feito isto, seu relatório está pronto. Deverá aparecer para você desta forma:



O relatório gerado sairá desta forma:

Data: 09:59:56
 Hora: 25/06/2009
 Página: 1


Relatório de Pessoas

» Informações da Pessoa

Id Pessoa: 22 **Nome:** Vinicius Marconi **Sexo:** Masculino

Data Nascimento: 4/3/1987

Observação: Este é o componente dbMemo, usado para inserir texto livre..

Cód Pessoa	Time Futebol
	Grêmio


Cód Pessoa: 789

» Informações da Pessoa

Id Pessoa: 27 **Nome:** João da Silva Gulari **Sexo:** Masculino

Data Nascimento: 2/5/1960

Observação: Observação Observação Observação Observação Observação Observação Observação Observação Observação Observação
 Observação Observação Observação Observação Observação Observação Observação Observação Observação Observação
 Observação

Cód Pessoa	Time Futebol
	Grêmio


Cód Pessoa: 27

» Informações da Pessoa

Id Pessoa: 29 **Nome:** José da Silva **Sexo:** Masculino

Data Nascimento: 30/6/2009

Observação: vbasdhdbsalcbasioldac dascdasendsiçacdsa cdasmckdismakodsacd]-samodsacdsmkackdmaac dsamodsklamods
 cde]amcmclascd ascmdasmclçdasmilçmliçda odascdasmilçmliçmliçd sacdsamliçmliçdasodsacd]mliçasmliçdascmliçdasmc]mliçdasc
 d]ascmdkascdaodsacdsacdasca

Cód Pessoa	Time Futebol
	Grêmio

Cód Pessoa: 29

» Somatório de Master Fields

Nome: João da Silva Gulari (1)

www.pyhed.com

Parabéns ! Você construiu seu primeiro relatório.