

Using pyHed reports.

In this documentation we will to learn build report's using frameCustomRptCondition. This frame was developed for that programers can build report's with ease, speed and reliability.

In this document we not will detail the procedure of build forms (File build => frameMain insert => action Insert), we consider if you is build a report, you know a basic procedure. Then we will build the file. Exists two methods that is very important on report build. Are its: `__init__` and `evtBeforeBuild`. Now, we will detail these.

1° Step - `__init__()`: Is on this method that we will set MasterFields, DatalFields and the SQL report. See the example below.

```
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'

```

Now we will know all parameters:

- **self.reportPath:** Should be informed the full path and file name. Ex: C:\reports\myReport.pdf;
- **exportsFields:** List with tuples, as this default ['FIELD', 'LABEL'], where:
 - FIELD: Is the field name on query;
 - LABEL: Label that will appear on report.
- **listMasterFields:** List/Tuple whit the order of report MasterFields;
- **listDetailFields:** List/Tuple with the order of report DetailFields;
- **self.reportFields:** Defines the Master and Detail Fields of report. Note that is a list witch tuples, the first value of each tuple shouldn't be changed, only informing the second value. The segund value on list is a list, like these showed behind.

Now, we can execute the `__init__` of inherits informing the necessary parameters. See the example below:

```

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

```

After the `__init__` two more parameters are informed:

- **self.SQL:** Should be informed the report SQL. Important: Always must be informed the `%WHERECLAUSE%` clause, its will be replaced by user clause.
Obs.: On future the pyHed will allow the programmers to inform the ORM class.;
- **self.title:** Application title.

2° Step - `evtBeforeBuild(self, report)`: This event, like its name say, is executed before build the report. Have only a parameter, the report instance, from this parameter we have full access to inherited proprieties. Its become the report framework more flexible. Follows the properties:

- **MasterColumnsMemo:** MasterField Columns, its text is a Memo;
- **Encoding:** Report encoding, used to convert texts;
- **SavePath:** Path and name of PDF file;
- **ImagePath:** Path and name of header image;
- **QtyColumnsMaster:** Number of items on header;
- **MasterKey:** Report primary key, Is responsible for separates the registers on report;
- **HeadersMaster:** MasterFields labels, if it be separated;
- **FieldsHeadersMaster:** Fields that belong for each header label;
- **ReportHeaders:** Titles like date, hour and page on header;
- **TotalizerLabels:** Text that will be show on occurrence sum of word existing on masters fields;
- **TotalizerColumnSum:** Text that will be show on occurrence sum of word existing on masters details;
- **TotalizerColumnCount:** Text that will be show on occurrence general columns sum on details;
- **DetailRegistersSum:** Sum columns defines for each register;
- **MasterColumnsCounter:** Sum occurrence word on MasterFields;
- **Title:** Report title, localized on header;
- **Footer:** Footer report;
- **DataSet:** Array that contains search registers;
- **DetailsCounters:** Fields for occurrence fields of one or more words on detail, show on report end;
- **DetailsSum:** Fields for sum of columns values on details. Show on report end;
- **DetailsCodeBar:** Code bar details (If exists);
- **DetailColumnGroup:** Detail columns that group by values on registers;
- **ReportFields:** Property that defines Master/Detail;
- **FlagHeader:** Flag that defines the display of header;
- **FlagFooter:** Flag that defines the display of footer.
- **FlagCounter:** Flag that defines the display of counters sum.

Now we know the properties that we will see on code example.

```

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}

```

3° Step – onPaint(): Is on this method that we will declare the components that will make the filters on report generate. This components from condComponents library, It was build for generate filters on report, only create its, your report will make filters automatically. Follows code example:

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

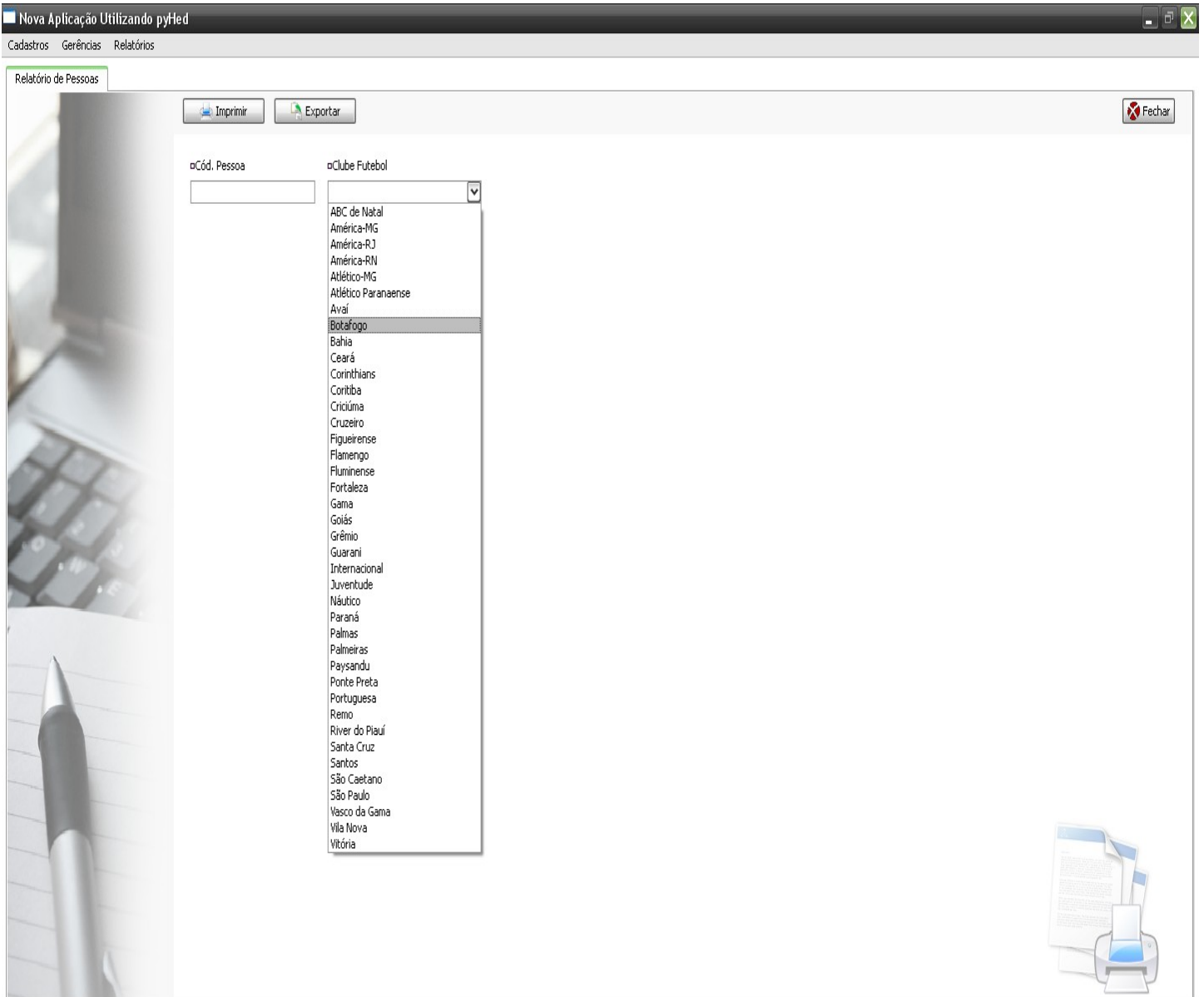
    self.edtNro = condComponents.CondNumericEdit(self.PnlMain,
                                                condition="P.COD_PESSOA = %VALUE%",
                                                required=False, text=u'&Cód. Pessoa',
                                                x=20,
                                                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 that the only difference on this components is the parameter “condition”, in this parameter inform the filter that be used. The filter must be equal as SQL filters, replaced the value by “%VALUE%”. On run time its will be replaced.

OK, your report is ready. Must be appear like this:



The generated report must be like this:

