

Neste exemplo vamos construir uma tela que herdará o `frameCustomSearcher`. Este frame foi construído para permitir ao programador, com poucas linhas de código ter uma tela de gerência realizando pesquisas conforme filtros informados e exportando os resultados para CSV automaticamente.

Tem somente um parâmetro de inicialização “**params**” que por default é nulo. Será através dele que vamos passar os parâmetros para inicialização da nossa tela.

São necessários 3 passos para construir sua tela:

- Construir o arquivo;
- Adicioná-lo ao `frameMain`;
- Adicionar a ação no banco.

1º Passo – Criando o arquivo: Crie um arquivo dentro da pasta “desktop”, neste exemplo como será uma gerência de pessoas vamos chamá-lo de “`frameGerenciaPessoas`” seguindo o padrão já usado na aplicação. Este é o arquivo:

```

#-*- encoding: iso-8859-1 -*-
#-*- coding: iso-8859-1 -*-

import sys
import PyQt4
import appConsts
sys.path.append(appConsts.pyHedPath)
sys.path.append(appConsts.appDbPath)
from pyHed.common import *
from pyHed.components import *
from pyHed.frames import *
import frameCadPessoa

class FrameGerenciaPessoas(frameCustomSearcher.FrameCustomSearcher):
    def __init__(self, parent, params=None):
        # Configura a Gerência
        params = {}
        params['Caption'] = u'Gerência de Pessoas'
        # sql de pesquisa
        params['SearchSQL'] = u"""
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
from
    PESSOA P
where
    %WHERECLAUSE% and
    ID_PESSOA <> 1
        """

```

```

# Título das colunas no grid de resultado
params['SearchColumnsTitle'] = ['*ID_PESSOA',
                                u'Código Pessoa',
                                'Nome',
                                'Sexo',
                                'Data Nascimento']

# campos para pesquisa
params['SearchFields'] = [
    (u'Código, P.COD_PESSOA, integer'),
    (u'Nome, P.NOME, string'),
    (u'Sexo, P.SEXO, string')
]
# primary key (campo que será usado na passagem para uma outra tela)
params['PrimaryKey'] = 'ID_PESSOA'

super(FrameGerenciaPessoas, self).__init__(parent, params, False, True)
self.title = u'Gerência de Pessoas'

def onPaint(self):
    super(FrameGerenciaPessoas, self).onPaint()

def evtGridResultDoubleClicked(self, aRow, aCol):
    self.setCursor(PyQt4.QtCore.Qt.WaitCursor)
    try:
        pyHedConsts.frmMain.showFrame(frameCadPessoa.FrameCadPessoa,
                                       self,
                                       ['GERENCIA_PESSOAS',
                                       int(self.gridResult.item(aRow, 0).text())])
    finally:
        self.setCursor(PyQt4.QtCore.Qt.ArrowCursor)
  
```

Agora vamos conhecer os parâmetros:

- **params['Caption']**: Deve ser informado o título da tela;
- **params['SearchSQL']**: Esta é a sql que será executada na pesquisa da tela, sempre deve conter a Primary Key da tabela e os campos que deseja mostrar no grid de resultados. 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.;
- **params['SearchColumnsTitle']**: É uma lista com os títulos das colunas no grid de resultados da pesquisa. Note que o primeiro é a chave primária da tabela antecedida por um *, isto significa que ela será uma coluna escondida que conterá a chave primária para passar para a tela de cadastro quando houver a seleção. Note que a ordem das colunas na lista é a mesma em que os campos se dispõem na select de pesquisa, tome sempre o cuidado de manter esta ordem, caso contrário não funcionará como esperado.
- **params['SearchFields']**: É uma lista com os filtros que o usuário poderá utilizar para

realizar a pesquisa. Cada item deve ser uma string separada por vírgulas que deve conter na seguinte ordem: “*label, CAMPO, tipo*”. Onde:

- **label**: Texto que será visível ao usuário;
- **campo**: a clausula que será inserida
Obs.: se a sua select possuir algum 'join' e as tabelas possuírem alias por exemplo PESSOA P o valor na string deve ficar P.NOME seguindo o exemplo acima. Isto evita problemas com colunas ambíguas;
- **tipo**: É o tipo do campo.
 - Tipos permitidos: string, integer, date;
- **params[**SearchFields**]**: Deve ser informado a chave primária da tabela ou view em questão.

Pronto o params já tem tudo o que precisa. Agora basta executar o `__init__` da herança como no exemplo abaixo.

```
super(FrameGerenciaPessoas, self).__init__(parent, params, False, True)
```

Método onPaint()

Neste exemplo só vamos reescreve-lo e chamar a herança. Mas ele pode ser usado para adicionar componentes como botões para outras telas na barra à esquerda. Segue o código do nosso exemplo:

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

Método “evtGridResultDoubleClicked”

Este é método que é atribuído ao clique duplo do grid. Se a sua tela de gerência não deve executar nenhuma ação no clique duplo do grid, reescreva o método com “pass” para que não execute nenhuma ação. Segue exemplo:

```
.....  
def evtGridResultDoubleClicked(self, aRow, aCol):  
    pass  
.....
```

No nosso exemplo vamos abrir a tela de cadastro a partir do registro selecionado no grid, então construiremos o evento da seguinte forma:

```
def evtGridResultDoubleClicked(self, aRow, aCol):
    self.setCursor(PyQt4.QtCore.Qt.WaitCursor)
    try:
        pyHedConsts.frmMain.showFrame(frameCadPessoa.FrameCadPessoa,
                                     self,
                                     ['GERENCIA_PESSOAS',
                                      int(self.gridResult.item(aRow, 0).text())])
    finally:
        self.setCursor(PyQt4.QtCore.Qt.ArrowCursor)
```

Desta forma abriremos a tela de cadastro a partir do item selecionado no grid.

Pronto! Sua tela está OK. Agora vamos criar o menu no frameMain da aplicação.

Adicione o menu ao frameMain como o exemplo abaixo.

```

# Import dos Frames de Cadastro do Sistema
import frameCadPessoa
import frameGerenciaPessoas
import frameRelatorioPessoas

# Main class
class FrameNovaAppMain(frmCustomMain.FrmCustomMain):
    def __init__(self):
        # Login and start app
        super(FrameNovaAppMain, self).__init__(frameLogin.FrameLoginNovaApp,
                                                '%s/splashScreen.png' % appConsts.appImagePath,
                                                windowMaximize=False)

        super(FrameNovaAppMain, self).setIcon('%s/appIco.ico' % appConsts.appImagePath)

    def registerFrames(self):
        """ frames registration """
        # Cria o menu.
        menuCadastro = self.addMenu('&Cadastros')
        # Cria o Item de Menu e atribui a ação a self.actCadPessoa
        # self.addMenuItem(nome do menu, Classe, 'Nome da tela')
        self.actCadPessoa = self.addMenuItem(menuCadastro,
                                                frameCadPessoa.FrameCadPessoa,
                                                u'Cadastro de Pessoas')

        # Create menu
        menuGerencia = self.addMenu(u'&Gerências')
        # Create menu item
        self.actGerenciaPessoa = self.addMenuItem(menuGerencia,
                                                    frameGerenciaPessoas.FrameGerenciaPessoas,
                                                    u'Gerência de Pessoas')

```

Agora insira a ação, segue o exemplo:

```

insert into ACAO (ID_ACAO, ID_ACAO_PAI, NOME, CAPTION,
                DATA_CADASTRO, NRO_EXECUCAO, ARQUIVO, ATALHO)
values (
    gen_id(ACAO_SEQ, 1), --- ID da ação, buscar da sequence
    1, -- ID da ação pai, neste caso a ação ROOT
    'FRAMEGERENCIAPESSOAS', -- O nome do frame em caixa alta
    'Gerência de Pessoas', -- Título para a tela.
    current_date, -- Data de cadastro da ação
    0, -- Nº de execuções
    null, -- Arquivo (Não utilizado neste exemplo)
    null) -- Atalho (Não utilizado neste exemplo)

```

Pronto sua tela está pronta. Deverá aparecer desta forma.

Nova Aplicação Utilizando pyHed

Cadastros Gerências Relatórios

Gerência de Pessoas

Adicionar condição Remover condição Pesquisar Fechar

Campo Operação Condição
 Nome Contendo a

Filtros informados

Exibindo todos os registros

	Código Pessoa	Nome	Sexo	Data Nascimento
1	789	Vinicius Marconi	Masculino	4/3/1987
2	27	João da Silva Gulari	Masculino	2/5/1960
3	28	Rodrigo Jesuino da Silva	Masculino	2/8/2003
4	29	José da Silva	Masculino	30/6/2009

Parabéns! Você construiu uma tela de gerência.