Controlleurs

Table des matières

1 Introduction

Il s'agit de développer une interface cliente dialoguant avec un serveur spécifique. Cette interface devra également jouer le rôle de serveur via l'interface CORBA.

2 Interface idl

Le nouveau controlleur hérite de la classe *Dash : :Controller*.

```
#include "Dash.idl"

module NouveauModule
{
  enum ErrorType { Timeout, CmdError, ConnectionError, EmergencyStop };

  exception OnlinecalibratorError {
    ErrorType type;
    string message;
  };

  interface Controller: Dash::Controller
  {
    void nouvelleFonction(in short value) raises (OnlinecalibratorError);
    ...
  };
};
```

3 Contrôleur

Par contrôleur on entend :

- un client d'une appliquation.
- un serveur CORBA

```
#!/usr/bin/python
import os,sys
import string
import time

# imports generic
import Dash, Dash__POA
from dash.python import StateController

# imports specific
import NouveauModule, NouveauModule__POA
import tcpClient

# Attention: la place du 1er héritier n'est pas annodine !!
class NouveauModuleController_i(NouveauModule__POA.Controller,
                                StateController.StateController_i)
                                ...):

    def __init__(self, name, ...):
        StateController.StateController_i.__init__(self,name,
                            options=["-ORBclientCallTimeOutPeriod","600000"])
        ...
        self.stop = 0

    def MainLoop(self):
        while not self.stop:
            time.sleep(0.1)

    def nouvelleFonction(self, value):
        ...mon code de connexion au serveur C...


if __name__ == '__main__':

    # naming service's name given by `nameclt list LOGIN`
    name = "NouveauModule/Controller"

    c = NouveauModuleController_i(name, ...)
    c.MainLoop()
```

4 Client pour les tests

Il s'agit juste d'un client CORBA. Le serveur CORBA étant lui le contrôleur décrit juste au dessus.

Par client on entend :

- un client CORBA

```
#!/usr/bin/env python

import os, sys
from dash.python import StateController,Server,Message,LoopHandler
import Dash, Dash__POA
import NouveauModule, NouveauModule__POA
```

```
class NouveauModuleUIController(StateController.StateController_i):

    def __init__(self,
                 name = "NouveauModule/GUI",
                 options = ["-ORBclientCallTimeOutPeriod","600000"]):

        StateController.StateController_i.__init__(self,name = name,
                                                   options = options)


class NouveauModuleUI:

    def __init__(self,name = "NouveauModule/GUI"):

        self.controller = NouveauModuleUIController(name = name)
        self.find_controller()

    def find_controller(self):
        """Try to find the controller """
        name = "NouveauModule/Controller"

        self.server = self.controller.find_server(name)
        if self.server is not None:
            self.server = self.server._narrow(NouveauModule.Controller)


if __name__ == '__main__':

    ui = NouveauModuleUI()

    if ui.server == None:
        print "cannot contact Corba server"

    else:
        ui.server.nouvelleFonction(0xab)

    print "Terminating"
    ui.controller.destruct()
```

5 Makefile

```
.PHONY: all clean

all:
        omniidl -bpython -I/usr/local/hess/dash/idl idl/NouveauModule.idl

clean:
        @rm -fr NouveauModule
        @rm -fr NouveauModule_idl.py
        @rm -fr NouveauModule__POA
```

6 Graphisme

6.1 Client graphique

Par client graphique on entend :

- un client CORBA
- une interface graphique
- un serveur CORBA

Attention, il s'agit d'un client CORBA capable de joindre le controlleur (au même titre que le client décrit ci-dessus). Cependant ce client est par ailleurs lui même un serveur CORBA vis à vis de l'interface graphique de plus haut niveau (l'interface *caméra contrôle*.

Déroulement des appels pour les actions de l'automate compris dans le contrôleur.

```
Client                                          Serveur

NouveauModuleGUI::__init__()
Dash::StateController_i::Configure()
Dash::Controller_i::Configuring()
NouveauModuleController::Configuring()
NouveauModuleGUI::Configuring()  ------->  Dash::StateController::Configure()
                                           Dash::Controller_i::Configuring()
                                           Dash::Controller_i::CheckHardware()
                                           NouveauModule::Controller_i::ConfigureHardware()
```

Voir par exemple le code de ce client graphique : *guiClientCorba.py*

6.2 Controlleur graphique

Si le conrôleur (premier serveur CORBA) et le client (premier client CORBA) sont tous les deux écrits dans le même langage, alors on écrira directement leur code au sein d'un même programme afin de se passer d'un appel au bus CORBA.

Par contrôleur on entend à présent :

- un client d'une appliquation.
- une interface graphique
- un serveur CORBA

Voir par exemple le code suivant : *guiControleurCorba.py*

Voici un DIFF soulignant les principales différences avec le code du client graphique donné ci-dessus :

```
37c39,41
< class OnlinecalibratorGUIController(StateController.StateController_i):
---
> class OnlinecalibratorGUIController(Onlinecalibrator__POA.Controller, # Attention!! : l'ordre est
>                                     StateController.StateController_i,
>                                     tcpClient.LedEmbeddedClientSocket):

43a44
>         tcpClient.LedEmbeddedClientSocket.__init__(self)

47,52d55
```

```
<     def find_server(self,name):
<         server = Server.Server_i.find_server(self,name)
<         if server is None: return server
<         server = server._narrow(Dash.StateController)
<         return server
<

95a98,114
>     def doDelay(self, valueOn2bytes):
>         self.SendDelay(valueOn2bytes)
>     ...
>

104c123
<         self.server.doDelay(string.atoi(hexaValue, 16))
---
>         self.controller.doDelay(string.atoi(hexaValue, 16))
...

128c147
<     def __init__(self,name = "Onlinecalibrator/GUI"):
---
>     def __init__(self,name = "Onlinecalibrator/Controller"):

132,133c151,152
<         self.set_wmclass('hess_main', 'Onlinecalibrator GUI')
<         self.set_title('Onlinecalibrator GUI')
---
>         self.set_wmclass('hess_main', 'Onlinecalibrator Controller')
>         self.set_title('Onlinecalibrator Controller')

147c166
<         self.onlinecalibratorframe = gtk.Frame(label = "Onlinecalibrator/GUI")
---
>         self.onlinecalibratorframe = gtk.Frame(label = "Onlinecalibrator Controller")

150,151d168
<         self.onlinecalibratorversion = gtk.Label("Controller not found")
<         self.onlinecalibratorversion.set_name("bold20")

155d171
<         h.attach(self.onlinecalibratorversion,0,2,0,1,gtk.EXPAND|gtk.FILL,gtk.EXPAND|gtk.FILL,5,5)

247,249d262
<
<         self.find_controller()
<

255,257c269,270
<             self.find_controller()
<             if self.server.GetState() == Dash.StateController.Safe:
<                 self.server.Configure()
---
>             if self.controller.GetState() == Dash.StateController.Safe:
```

```
>                 self.controller.Configure()

259,260c272
<             self.add_command_bunch(c,self.onlinecalibratorstatus.set_text,‘self.server.GetState()‘
---
>             self.add_command_bunch(c,self.onlinecalibratorstatus.set_text,‘self.controller.GetStat

265c277
<             self.server = None
---
>             self.controller = None

271d282
<             self.find_controller()
...

363,381d364
<
<     def find_controller(self):
<         """Try to find the controller """
<         name = "Onlinecalibrator/Controller"
<
<         self.server = self.controller.find_server(name)
<
<         if self.server is not None:
<             self.server = self.server._narrow(Onlinecalibrator.Controller)
<
<         c = self.create_command_bunch()
<
<         if self.server is None:
<             self.add_command_bunch(c,self.onlinecalibratorstatus.set_text,"N/A")
<             self.add_command_bunch(c,self.onlinecalibratorversion.set_text,"Cannot find object Onl
<         else :
<             self.add_command_bunch(c,self.onlinecalibratorstatus.set_text,‘self.server.GetState()‘
<             #self.add_command_bunch(c,self.onlinecalibratorversion.set_text,self.server.GetControl
<         self.execute_command_bunch(c)
435a419
>
```