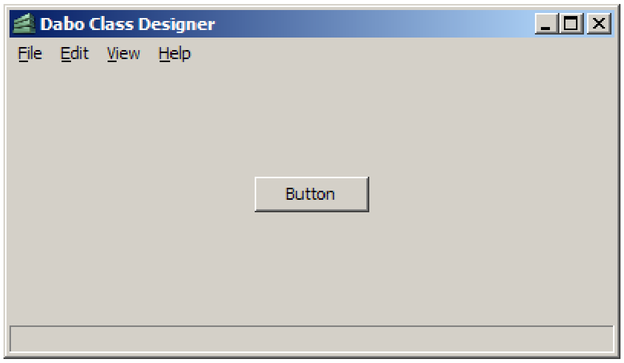
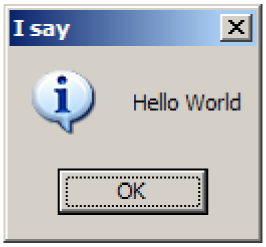
by Bill Maxwell
bill_maxwell_nospam@comcast.net

Dabo Tutorial: Hello_msgBox

Using the Dabo Class Designer to produce what may be the simplest GUI application.





Hello World

Document Version: V1.02, 18 July 2007

Overview

This tutorial shows how to create a very simple Graphical User Interface (GUI) application using the Dabo Class Designer.

We'll create a "Hello World" program that uses a single user interface object (a button) and a single line of code, to open a standard information message box that says hello. This may very well be the simplest possible interactive GUI application that you can create using the visual design tools in Dabo.

This tutorial assumes you are using Windows, but the procedure should be very similar on other supported systems.

> **Note:** This tutorial assumes you have installed the Dabo and Dabo IDE packages, and the things needed to make them run properly. In addition to the Dabo packages, you will need either the Dabo runtime binary package, or both Python and wxPython. You can learn how to do that here: http://dabodev.com/.

This was written using screenshots from these software versions:

Platform: Windows XP (with Windows Classic theme)
Python Version: 2.5 on win32
wxPython Version: 2.8.3.0 on win32
Dabo Version: Version 0.7.2s; Revision 2719
Dabo IDE version: 0.7.1s (Revision 755)

However, this tutorial is simple and basic enough that things should not change much for newer versions of the above software.

Create the application

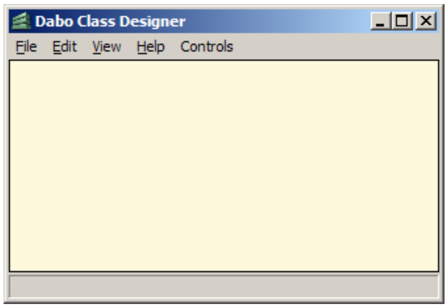
Run ClassDesigner:

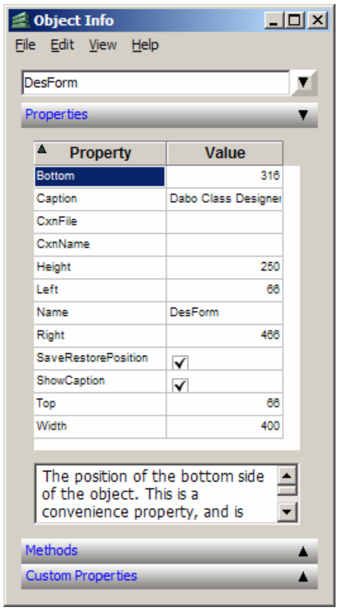
On many systems, you can just double-click this file to start Class Designer:

…\Python\Lib\site-packages\daboide\ClassDesigner.py

This opens the following windows:

1. Dabo Class Designer main window (used to design the user interface).
2. Object window (shows properties of object selected in main window).
3. An Editing window (used to enter code attached to user interface objects).
4. A Python window that shows status and error messages.

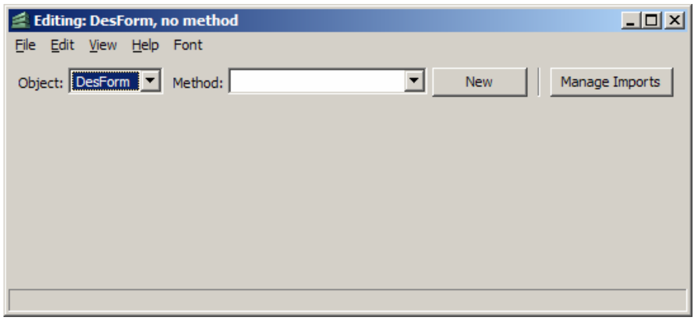




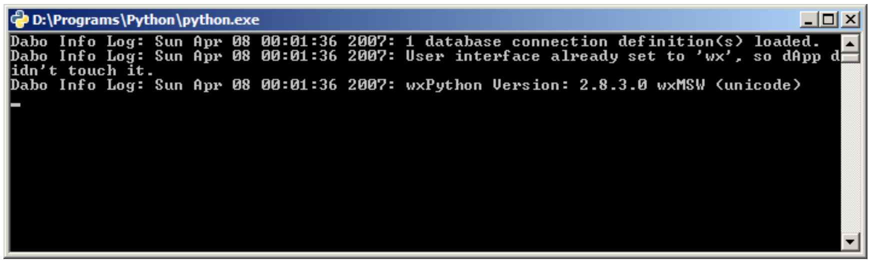
DesForm

Property	Value
Bottom	316
Caption	Dabo Class Designer
CxnFile	
CxnName	
Height	250
Left	66
Name	DesForm
Right	466
SaveRestorePosition	☑
ShowCaption	☑
Top	66
Width	400

The position of the bottom side of the object. This is a convenience property, and is



Object: DesForm Method:



```
Dabo Info Log: Sun Apr 08 00:01:36 2007: 1 database connection definition(s) loaded.
Dabo Info Log: Sun Apr 08 00:01:36 2007: User interface already set to 'wx', so dApp d
idn't touch it.
Dabo Info Log: Sun Apr 08 00:01:36 2007: wxPython Version: 2.8.3.0 wxMSW (unicode)
_
```

Save the blank application.

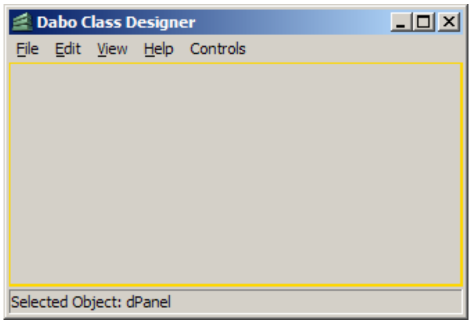
1. Hit Ctrl-S, or use the File / Save menu
2. Name the application "Hello_msgBox".
3. This creates these files:
   - Hello_msgBox.cdxml (A file that contains the GUI design for your app.)
   - Hello_msgBox-code.py (Contains custom code attached to GUI objects.)

Add a panel object:

In Windows, you need to add a panel to contain all the controls in the current form. This seems to be necessary so that the form picks up any Windows Theme information and correctly displays the background. In other operating systems, this step isn't necessary (but it's good practice anyway, so your applications can be used in all places).

1. Right-click in the blank area of the Dabo Class Designer screen.
2. Use the Display Controls > Add Panel menu to add a panel to the blank slot.

3. The Object Info window should now show "dPanel" when you select the panel you just added.

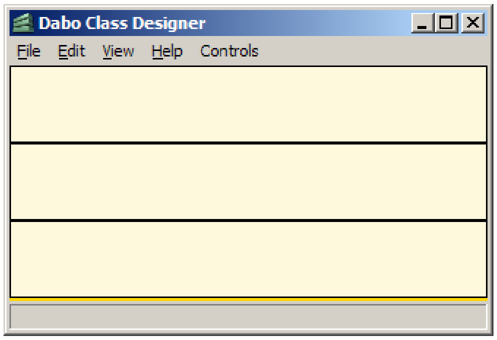


Selected Object: dPanel

Add a sizer to the panel:

A sizer can be thought of as an automatically resizing set of rows, or columns, or a grid, that you put inside of a container. When the container is resized, the sizer automatically changes size, and resizes the slots within the sizer according to rules that you set.

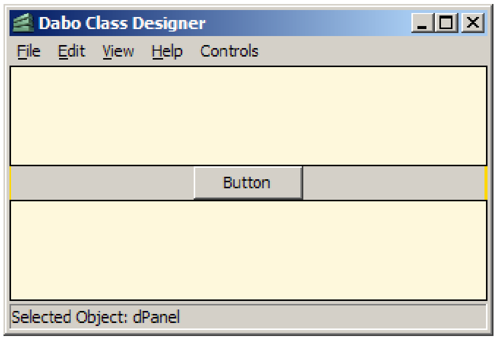
1. Right-click in the blank area where you just added the panel.
2. Use the Sizers / Add New Vertical Sizer menu, and set number of slots to 3.



We create 3 slots, because we will put a button in the middle slot and leave the other two slots empty, which vertically centers the button.

Add a button:

1. Click in the middle slot of the sizer to select it.
2. Right click in the sizer, and use the Interactive Controls / Add Button menu.



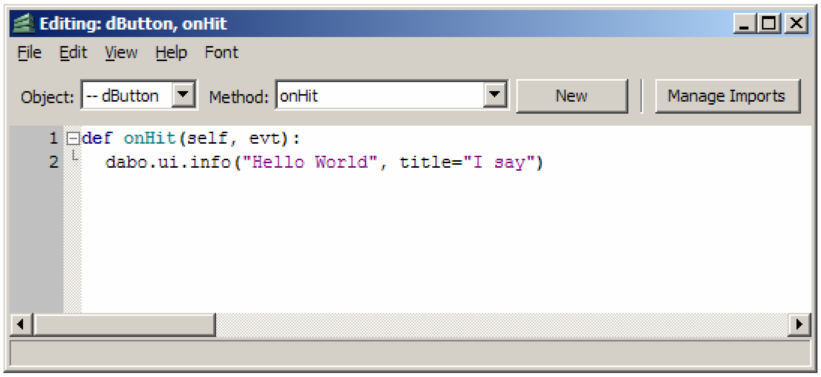
Selected Object: dPanel

Add code to the button:

1. Right-click the button and choose Edit Code.
2. This brings up the Dabo Editing window, with the dButton object selected.
3. Use the Method pull-down box to select the onHit method (which is called when you press the

button).

4. Enter this line of code in the onHit function:
   dabo.ui.info("Hello World", title="I say")
5. Save your code by hitting Ctrl-S, or using the File / Save menu.



```
def onHit(self, evt):
    dabo.ui.info("Hello World", title="I say")
```

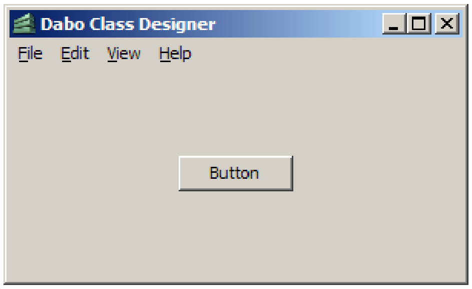
Save the overall application

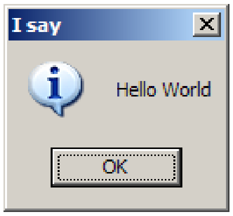
Select the Dabo Class Designer main window and hit Ctrl-S, or use the File / Save menu.

Run the application

Run the application using the main Class Designer window's File / Run menu. This runs the application, which is just a window with a button in the middle, labeled "Button".

Hit the button, and a standard information message box will pop up in the middle of your screen that says "Hello World", and has a title of "I say".





Hello World

Congratulations, you just wrote and ran a Dabo GUI application!

Exit your application using the "X" button, or File / Quit.

After you close the Dabo Class Designer, and later when you want to open the Hello_msgBox app again, just run the designer, and open this file: Hello_msgBox.cdxml. If you know your way around Windows, you can set the cdxml file extension so that you can double-click these files and automatically start the Dabo Class Designer.

Create a Runnable Python Application

Above, you ran your program from inside the Dabo Class Designer. Now, create a python program file that you can run outside the Designer:

- Use the Dabo Class Designer main window menu and choose File / Save Runnable App.
- This creates a file named "Hello_msgBox.py" which you can run the same way you run any other Python application.

Extra Credit

There are other types of message boxes that you can use instead of the “info” type. Try using these lines of code to see the other types:

- dabo.ui.exclaim("Hello World!", title="I exclaim")
- dabo.ui.stop("And Say Hello World.", title="Hey You")

Here’s another message box type of function that returns an answer from the user:

```
response = dabo.ui.areYouSure("Do you want to save your changes?",
                title="Unsaved Changes (Dabo Tutorial)")
if response == True:
        dabo.ui.info("Changes have been saved.", title="Action Taken (Dabo Tutorial)")
elif response == False:
        dabo.ui.info("Changes have NOT been saved.", title="Action Taken (Dabo Tutorial)")
else:
        dabo.ui.stop("Operation cancelled.", title="Action Taken (Dabo Tutorial)")
```

Also, if you play around a little bit, you can easily figure out how add a little polish to the application – by using the Object Info window to change the caption on the button, for example.

Conclusion

This tutorial intentionally tries to make the simplest possible GUI application using the Dabo Class Designer.

In this tutorial, we use a message box to show our “Hello World” output, because you can create a message box without having to worry about how one object can reference another. For example, if you wanted the button object, when pushed, to be able to write text into an editBox, then the button object would have to know how to reference the editBox object to put text into it. We leave that for another tutorial (Hello_editBox).

References

- Dabo Documentation: http://dabodev.com/documentation

- Dabo-Users List archives: http://leafe.com/archives/search/dabo-users