Workbench Scripting: Using Forms in your Script

by **Eric Miller** September 7, 2011 10:06 am **Leave a comment** **The Focus**

Last week we had a very well attended Webinar on using Scripting with Workbench. We talked about how you can control projects, parameters, and material properties with python. We also talked about how you can use python to drive data compliant applications by sending commands and/or parameters to it. You can watch the webinar on YouTube **here**.

Click on PADT ANSYS Webinar Series and then find the recording from August 11, 2011.

Several attendees asked if we had an example of using python to bring up a GUI on top of workbench. We did not. This is how The Focus articles are born.

The Basics

In case you missed the webinar or didn't follow some of it, here are the important points:

- Workbench has IronPython built into it from the ground up
- You can record whatever you do in the GUI to create scripts
- You can read those scripts in to control Workbench
- ANSYS uses IronPython because it talks to Windows programs
- Get detailed info at **www.python.org**, **www.ironpython.net**, and **www.ironpython.info**

Most of the examples below are derived from information I got at: **http://www.ironpython.info/index.php/Contents#Windows_Forms**

We will be making a GUI for the famous Tower of Test application. We will need a place to enter in the size of the tower, a pressure load. Then a simple table showing the results when we get them. And of course, buttons to cancel and execute.

If you are already an experienced Windows programmer, this will be a slam dunk. If you are not, then you will need to make sure you learn your way around the MSDN .Net Class Library Reference:

http://msdn.microsoft.com/en-US/library/ms229335(v=VS.80).aspx

That is where you find properties and methods for the GUI. We will be spending most of our time in System.Windows.Form.

Why .NET?

Right now, some of you might be asking "Why .NET?" Python supports probably over a dozen different GUI's. I am using .NET for a few reasons:

1. It is built into IronPython. No need to download a library and install it right
2. It will have a consistent look and feel to Workbench
3. More people know .NET than the other GUI libraries.

If you are anit-microsoft, you can poke around and find a library that fits your needs. The process will be the same.

Crawl: Always Start with Hello World

Anytime I try and learn a new language, I usually find a "Hello World" Example. And, thanks to the internet I found one that only needed slight modification:

```
import clr
clr.AddReference('System.Windows.Forms')

from System.Windows.Forms import Application, Form, Label

form = Form(Text="Hello World Form")
label = Label(Text="Hello World!")
```

Privacy - Terms

```

form.Controls.Add(label)

Form.ShowDialog(form)
```

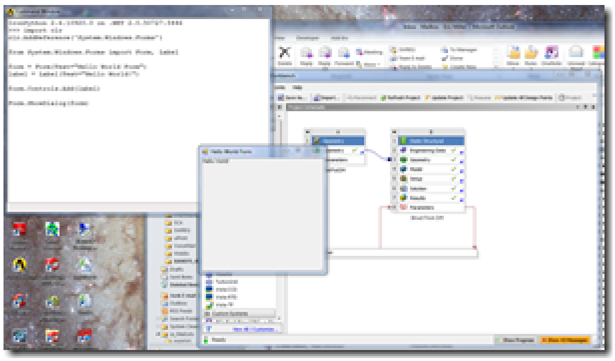
So first, we access the huge number of Windows tools by importing clr. Then, get a reference to Windows Forms. Once that is done we need to import more from Windows Forms. This is a big tool box so it has lots of levels you have to dig through. For this guy we need to be able to do the form itself (the window) and a label (text in the window).

Now that we have our objects, we create a form with the title “Hello World Form” and a label with the text “Hello World!” We then use the Add method to add the label control to the form. Last, we use the Show Dialog method to show the form. TADA.

To give it a try, copy and paste this into your own text editor and then go into Workbench and choose:

File->Scripting->Open Command Window…

Then paste in the script. Or you can save it to a file and read it in. I like to paste so I can see my error messages as it goes.



Now you are a Workbench GUI programmer. Put it on your resume!

Walk: Add Some Controls to the Window

Next we want to do a bit more than saying hello.

The important thing I learned that I just didn’t’ get before is that you have to make a class out of your form and any functions that get called when the user clicks on a button or interacts in any way with the form. You do this because Windows will then pass information about the form back to the event function.

In the example below is like “Hello World” to start, then we make 4 text boxes with labels for the user to enter data into. We then add an OK and a Cancel button. The code is commented with an explanation of what is going on. If something does not make sense, look it up in a python reference or ask someone who is a python programmer.

```
# --  Bring in the .NET stuff, and reference Forms and drawing
import clr
clr.AddReference('System.Windows.Forms')
clr.AddReference('System.Drawing')

#   get the objects you need from forms and drawing
from System.Windows.Forms import Form, Label, Button, TextBox
from System.Drawing import Point, ContentAlignment

#-- The way python works with events, you have to make a class out of your form.
#   Then the objects
#    you add are available in event functions.
class TowerForm(Form):
    def __init__(self):
#-- First, make the form (self)
        self.text="Tower Simulation Tool"
        self.Height=500
        self.Width=250

#-- Create a Simple Text Label describing the tool
#   it is not self.label because we are not going to access it in event functions
        lbl1 = Label()
        lbl1.Text = "This is the Tower of Test Tool. \n\
            Please Enter values and click OK to run the \
            model and get new results\n"
        lbl1.Height = 50
        lbl1.Width = 200

#-- Create the Text Labels and text boxes to get Dimensions and Pressure
```

```
#   We need to get to the textboxes so we will make them part of the class
#
#   some constants to make the positioning easier
        x1 = 10
        y1 = 80
        w1 = 70
        w2 = 80
        x2 = x1 + w1 + 10
#   Make the labels for the text boxes.  This is done in the Labe() call with
#   Name = Value arguments
#   instead of object.name = value to save space.  All in one line instead of
#   four lines.
#     The alighment is done so that the labels are all right justified to line
#     up with the boxes. Set ContentAlignment.MiddleRight to mr to save space
        mr = ContentAlignment.MiddleRight
        lb_Length=Label(Text="Length", Width=w1,TextAlign=mr)
        lb_Width  = Label(Text="Width",   Width = w1, TextAlign=mr)
        lb_Height = Label(Text="Height",  Width = w1, TextAlign=mr)
        lb_Press  = Label(Text="Pressure",Width = w1, TextAlign=mr)

#   Make the text boxes. Note that these are put in the self. class.
#   We do this so that when the OK
#   button is pushed, we have access to the text boxes and therefore the values
#   typed within

        self.tb_Length = TextBox(Width = w2)
        self.tb_Width  = TextBox(Width = w2)
        self.tb_Height = TextBox(Width = w2)
        self.tb_Press  = TextBox(Width = w2)

#   Specify the location for the label and the text boxes.  Move down by
#   30 after each line
        lb_Length.Location = Point(x1,y1)
        self.tb_Length.Location = Point(x2,y1)
        y1 = y1 + 30
        lb_Width.Location = Point(x1,y1)
        self.tb_Width.Location = Point(x2,y1)
        y1 = y1 + 30
        lb_Height.Location = Point(x1,y1)
        self.tb_Height.Location = Point(x2,y1)
        y1 = y1 + 30
        lb_Press.Location = Point(x1,y1)
        self.tb_Press.Location = Point(x2,y1)

#   Make an OK and a Cancel Button
        okBut = Button(Text="OK", Width=50)
        okBut.Location = Point(30,430)
#       This is where you specify the funtion to be called when the button is clicked
#       note that you call the function self.functionname.
        okBut.Click += self.okButPressed

        cancelBut = Button(Text="Cancel", Width=50)
        cancelBut.Location = Point(110,430)
        cancelBut.Click += self.cancelButPressed

#   Put everything on the form (some sort of list and loop would make this easier)
        self.Controls.Add(lbl1)
        self.Controls.Add(lb_Length)
        self.Controls.Add(self.tb_Length)
        self.Controls.Add(lb_Width)
        self.Controls.Add(self.tb_Width)
        self.Controls.Add(lb_Height)
        self.Controls.Add(self.tb_Height)
        self.Controls.Add(lb_Press)
        self.Controls.Add(self.tb_Press)
        self.Controls.Add(okBut)
```

```python
        self.Controls.Add(cancelBut)

#-- Now define the button event functions as part of the class
#
# Cancel simply closes the form down.  Nothing fancy
    def cancelButPressed(self, sender, args ):
        print 'Closing the Window... Bye...\n'
        self.Close()

# OK prints the values of the text boxes to the console
#   This will be replaced with calls to workbench for the next step
    def okButPressed(self, sender, args ):
        print 'Values Entered:\n'
        print '  Length: %s\n' %self.tb_Length.Text
        print '   Width: %s\n' %self.tb_Width.Text
        print '  Height: %s\n' %self.tb_Height.Text
        print 'Pressure: %s\n' %self.tb_Press.Text
        self.Close()
#---------End of class definition


# Instantiate the form and make the form visible
print '=======================================\n'
print 'Opening the Windows...\n'
myForm = TowerForm()
Form.ShowDialog(myForm)

```

If you have not programmed a go dialog box in a while by hand, instead of using a WYSIWIG GUI builder, you can see why. Lots of lines… lots of lines…
Here is the file without the numbers in it so you can run it in Workbench:
TowerToolTextOnly.py

Run: Send Values to Workbench, Update Model, Get Results

If you remember the previous article on **Driving Workbench from Excel** this will all look familiar. If you don't review the article, especially the part about getting parameters in and out of Workbench with Python. To finish our example we will use the same basic code to:

  - Get the current values from Workbench and set our dialog to show those values
  - When the user hits OK:
  - Set the Workbench parameters to the current dialog box parameters
  - Update the model
- Get the resulting deflection and display it.

The code is shown below, with the bits added for the Workbench interaction highlighted. Note that we also introduced the specification of font properties on a label. Making it even more fancy.

```
# --  Bring in the .NET stuff, and reference Forms and drawing
import clr
clr.AddReference('System.Windows.Forms')
clr.AddReference('System.Drawing')

#   get the objects you need from forms and drawing
from System.Windows.Forms import Form, Label, Button, TextBox
from System.Drawing import Point, ContentAlignment,
Font, FontStyle, GraphicsUnit

#-- The way python works with events, you have to make a class out of your form.
#   Then the objects
#    you add are available in event functions.
class TowerForm(Form):
    def __init__(self):
#-- First, make the form (self)
```

```
        self.text="Tower Simulation Tool"
        self.Height=500
        self.Width=250

#-- Create a Simple Text Label describing the tool
#   it is not self.label because we are not going to access it in event functions
        lbl1 = Label()
        lbl1.Text = "This is the Tower of Test Tool. \n\
            Please Enter values and click OK to run the \
            model and get new results\n"
        lbl1.Height = 50
        lbl1.Width = 200

#-- Create the Text Labels and text boxes to get Dimensions and Pressure
#   We need to get to the textboxes so we will make them part of the class
#
#   some constants to make the positioning easier
        x1 = 10
        y1 = 80
        w1 = 70
        w2 = 80
        x2 = x1 + w1 + 10
#   Make the labels for the text boxes.  This is done in the Labe() call with
#   Name = Value arguments
#   instead of object.name = value to save space.  All in one line instead of
#   four lines.
#     The alighment is done so that the labels are all right justified to line
#     up with the boxes. Set ContentAlignment.MiddleRight to mr to save space
        mr = ContentAlignment.MiddleRight
        lb_Length=Label(Text="Length", Width=w1,TextAlign=mr)
        lb_Width  = Label(Text="Width",   Width = w1, TextAlign=mr)
        lb_Height = Label(Text="Height",  Width = w1, TextAlign=mr)
        lb_Press  = Label(Text="Pressure",Width = w1, TextAlign=mr)
```

```

#   Make a label to give status of the update in:

self.updlbl = Label(Text=” “,Width=200, Height=100,\
TextAlign=ContentAlignment.BottomCenter)
self.updlbl.Location = Point(0,200)
self.updlbl.Font = \
Font(‘Verdana’,10, FontStyle.Bold, GraphicsUnit.Point,0)

#   Make the text boxes. Note that these are put in the self. class.
#   We do this so that when the OK
#   button is pushed, we have access to the text boxes and therefore the values
#   typed within

        self.tb_Length = TextBox(Width = w2)
        self.tb_Width  = TextBox(Width = w2)
        self.tb_Height = TextBox(Width = w2)
        self.tb_Press  = TextBox(Width = w2)

#   Specify the location for the label and the text boxes.  Move down by
#   30 after each line
        lb_Length.Location = Point(x1,y1)
        self.tb_Length.Location = Point(x2,y1)
        y1 = y1 + 30
        lb_Width.Location = Point(x1,y1)
        self.tb_Width.Location = Point(x2,y1)
        y1 = y1 + 30
        lb_Height.Location = Point(x1,y1)
        self.tb_Height.Location = Point(x2,y1)
        y1 = y1 + 30
        lb_Press.Location = Point(x1,y1)
        self.tb_Press.Location = Point(x2,y1)

        self.tb_Length.Text = Parameters.GetParameter(Name="P1").Expression
        self.tb_Width.Text = Parameters.GetParameter(Name="P2").Expression
        self.tb_Height.Text = Parameters.GetParameter(Name="P3").Expression
        self.tb_Press.Text = Parameters.GetParameter(Name="P5").Expression

#   Make an OK and a Cancel Button
        okBut = Button(Text="OK", Width=50)
        okBut.Location = Point(30,430)
#       This is where you specify the funtion to be called when the button is clicked
#       note that you call the function self.functionname.
        okBut.Click += self.okButPressed

        cancelBut = Button(Text="Cancel", Width=50)
        cancelBut.Location = Point(110,430)
        cancelBut.Click += self.cancelButPressed

#   Put everything on the form (some sort of list and loop would make this easier)
        self.Controls.Add(lbl1)
        self.Controls.Add(lb_Length)
        self.Controls.Add(self.tb_Length)
        self.Controls.Add(lb_Width)
        self.Controls.Add(self.tb_Width)
        self.Controls.Add(lb_Height)
        self.Controls.Add(self.tb_Height)
        self.Controls.Add(lb_Press)
        self.Controls.Add(self.tb_Press)

self.Controls.Add(self.updlbl)
        self.Controls.Add(okBut)
        self.Controls.Add(cancelBut)
```

```
#-- Now define the button event functions as part of the class
#
# Cancel simply closes the form down.  Nothing fancy
    def cancelButPressed(self, sender, args ):
        print 'Closing the Window... Bye...\n'
        self.Close()

# OK prints the values of the text boxes to the console
#   This will be replaced with calls to workbench for the next step
    def okButPressed(self, sender, args ):

# In Workbench: Get objects for all the parameters you need access to
#   Remember, it goes by parameter name, not the name you give them.

lenParam = Parameters.GetParameter(Name="P1")
widParam = Parameters.GetParameter(Name="P2")
hgtParam = Parameters.GetParameter(Name="P3")
prsParam = Parameters.GetParameter(Name="P5")
# don't forget the resulting deflection output parameter:

defParam = Parameters.GetParameter(Name="P4")

# Now set the workbench values to the values from the dialog

lenParam.Expression = self.tb_Length.Text
widParam.Expression = self.tb_Width.Text
hgtParam.Expression = self.tb_Height.Text

```

```
prsParam.Expression = self.tb_Press.Text

self.updlbl.Text = "Updating"


Update()

dflValue = defParam.Value.Value

self.updlbl.Text = "Deflection is %s in\n" %dflValue


#---------End of class definition


# Instantiate the form and make the form visible
print '======================================\n'
print 'Opening the Windows...\n'
myForm = TowerForm()
Form.ShowDialog(myForm)

```

Next Steps and Suggestions

And that in a nutshell is the basics of putting a GUI on Workbench. By using the calls to Workbench, .NET or any other complaint application, you can script away and do what you want.

Much more can be done. You could have the solver you are running in update and create some plots, and then use python to find those plots (say search for any *.jpg files in the project director created after the OK button was pushed) and insert them into PowerPoint with text that you also gather.

There really are no limitations.

To learn more, try stuff. Get to know the MSDN documentation to know what you can do to make a more professional GUI. The **Windows Forms part of ironpython.info** is a great resource for this and points to MSDN as needed.

There is a lot more you can do with ANSYS Workbench and the various solvers. And with each release, workbench will add and document more.

If you write a cool application you can share, make sure you put it on XANSYS or send it to Sheldon at ANSYS.net.

Related

Workbench and Excel, Part 2: Driving Workbench From Excel with Python
July 13, 2011
In "The Focus"

Starting ANSYS Products From the Command Line
February 29, 2012
In "The Focus"

Workshops for “Intro to Workbench Framework Scripting”
March 8, 2012
In "The Focus"

‹ Using the ‘Identifier’ for Mechanical Post-Processing

HSFLD241/242: Modeling Enclosed Liquids ›

Search

LINKS

PADT Website
Contact PADT
PADTMarket.com
Manage PADT Subscriptions

SUBSCRIBE TO OUR BLOG

Enter your email address to subscribe to this blog and receive notifications of new posts by email.

Join 1,804 other subscribers

Email Address

Subscribe

UPCOMMING EVENTS

01/19/2022 - 01/21/2022
Arizona Photonics Days

04/04/2022 - 04/07/2022
37th Space Symposium
Arizona Space Industry Booth

04/12/2022 - 04/14/2022
D&M West | MD&M West

3D PRINTING GLOSSARY

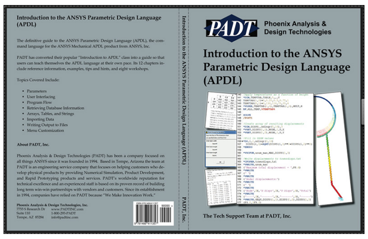
Privacy - Terms

3DP GLOSSARY

3D PRINTING GLOSSARY

A RESOURCE FOR ADDITIVE MANUFACTURING

INTRODUCTION TO THE ANSYS PARAMETRIC DESIGN LANGUAGE


PADT Phoenix Analysis & Design Technologies

Introduction to the ANSYS Parametric Design Language (APDL)

The Tech Support Team at PADT, Inc.

Learn APDL with this workshop based book written by PADT's Technical Support Team. The new and improved Second Edition contains additional chapters on APDL Math and APDL in ANSYS Mechanical.

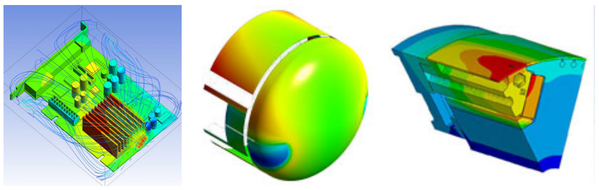
Now available on Kindle as well as paperback.

More…

SIMULATION SERVICES

PADT's simulation engineers are true experts in virtual prototyping. Trust the people you come to for ANSYS expertise to handle your simulation outsourcing needs.

Structural, Thermal, Fluid, Electromagnetic, and Systems.



LOOKING FOR ANSYS TRAINING?

Let the people who write “The Focus” train you and your team. Check out our schedule or contact us to set up a class at your place or something custom.

PODCAST: ALL THINGS ANSYS

All Things ANSYS Podcast

Presented by PADT

ANSYS ACADEMIC PROGRAM

Ansys ACADEMIC PROGRAM

Learn about software for students and research

RSS SUBSCRIBE:

RECENT POSTS:

- Surprise – 2021 Turned out to Be a Lot Like 2020
- All Things Ansys 102: Electronics Reliability Updates in Ansys 2021 R2
- Simulating an Electro-Permanent Magnet (EPM) using Ansys Maxwell

Ansys Pro – Premium – Enterprise Electronics Licensing Adjustments

All Things Ansys 101: Additive & Structural Optimization Updates in Ansys 2021 R2

CATEGORIES

Additive Manufacturing

Ansys

ANSYS Discovery

ANSYS Energy Innovation Campaign

ANSYS R 18

Carbon

Education

Events

Flownex

Fun

Getting To Know PADT

News

Nimbix

PADT Medical

PADT Startup Spotlight

Podcast

Product Development

Publications

Startups

Stratasys

Stratasys Marketing

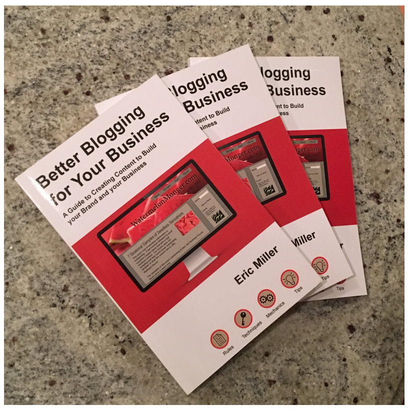
The Focus

Uncategorized

Webinar

Contact PADT
1-800-293-PADT
info@padtinc.com

THINKING ABOUT A BLOG FOR YOUR COMPANY?


Better Blogging for Your Business
A Guide to Creating Content to Build your Brand and your Business
Eric Miller

Where do you start? How do you keep it going? Where do I get ideas for posts? Should I use humor?

These questions and many others are answered in the book that was inspired by the success of PADT's blog:

Better Blogging for your Business

Available now as a Kindle book or softcover on Amazon.

PADT EMAIL

Manage how PADT Emails You

Want to receive Emails from PADT? Please select any of the basic topics below to tell us what you are interested in. We promise to keep it simple sharing news, opportunities, and useful information. You can come back and change your settings whenever you need

* Email

State/Province

Company

* Email Lists

- ☐ PADT Additive & Advanced Manufacturing Email List
- ☐ PADT General Information Email List
- ☐ PADT Product Development and Testing Email List
- ☐ PADT Simulation Email List

Sign Up!

Contact PADT
1-800-293-PADT
info@padtinc.com

SEARCH

Search

LINKS

PADT Website
Contact PADT
PADTMarket.com
Manage PADT Subscriptions

ANSYS STARTUP PROGRAM

LOOKING FOR ANSYS TRAINING?

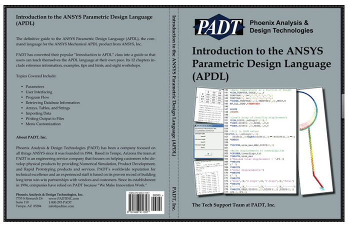
Let the people who write "The Focus" train you and your team. Check out our **schedule** or **contact us** to set up a class at your place or something custom.

TRYING TO FIND A COMPUTER BUILT FOR SIMULATION?

Workstations, Servers, and Clusters designed by PADT specifically for simulation users.

Learn More

INTRODUCTION TO THE ANSYS PARAMETRIC DESIGN LANGUAGE



PADT Phoenix Analysis & Design Technologies

Introduction to the ANSYS Parametric Design Language (APDL)

The Tech Support Team at PADT, Inc.

Learn APDL with this workshop based book written by PADT's Technical Support Team.

The new and improved Second Edition contains additional chapters on APDL Math and APDL in ANSYS Mechanical.

Now available on Kindle as well as paperback

More...

RECENT POSTS

- Surprise – 2021 Turned out to Be a Lot Like 2020
- All Things Ansys 102: Electronics Reliability Updates in Ansys 2021 R2
- Simulating an Electro-Permanent Magnet (EPM) using Ansys Maxwell
- Ansys Pro – Premium – Enterprise Electronics Licensing Adjustments
- All Things Ansys 101: Additive & Structural Optimization Updates in Ansys 2021 R2

CATEGORIES

- Additive Manufacturing
- Ansys
- ANSYS Discovery
- ANSYS Energy Innovation Campaign
- ANSYS R 18
- Carbon
- Education
- Events
- Flownex
- Fun

Getting To Know PADT

News

Nimbix

PADT Medical

PADT Startup Spotlight

Podcast

Product Development

Publications

Startups

Stratasys

Stratasys Marketing

The Focus

Uncategorized

Webinar

Connect with PADT