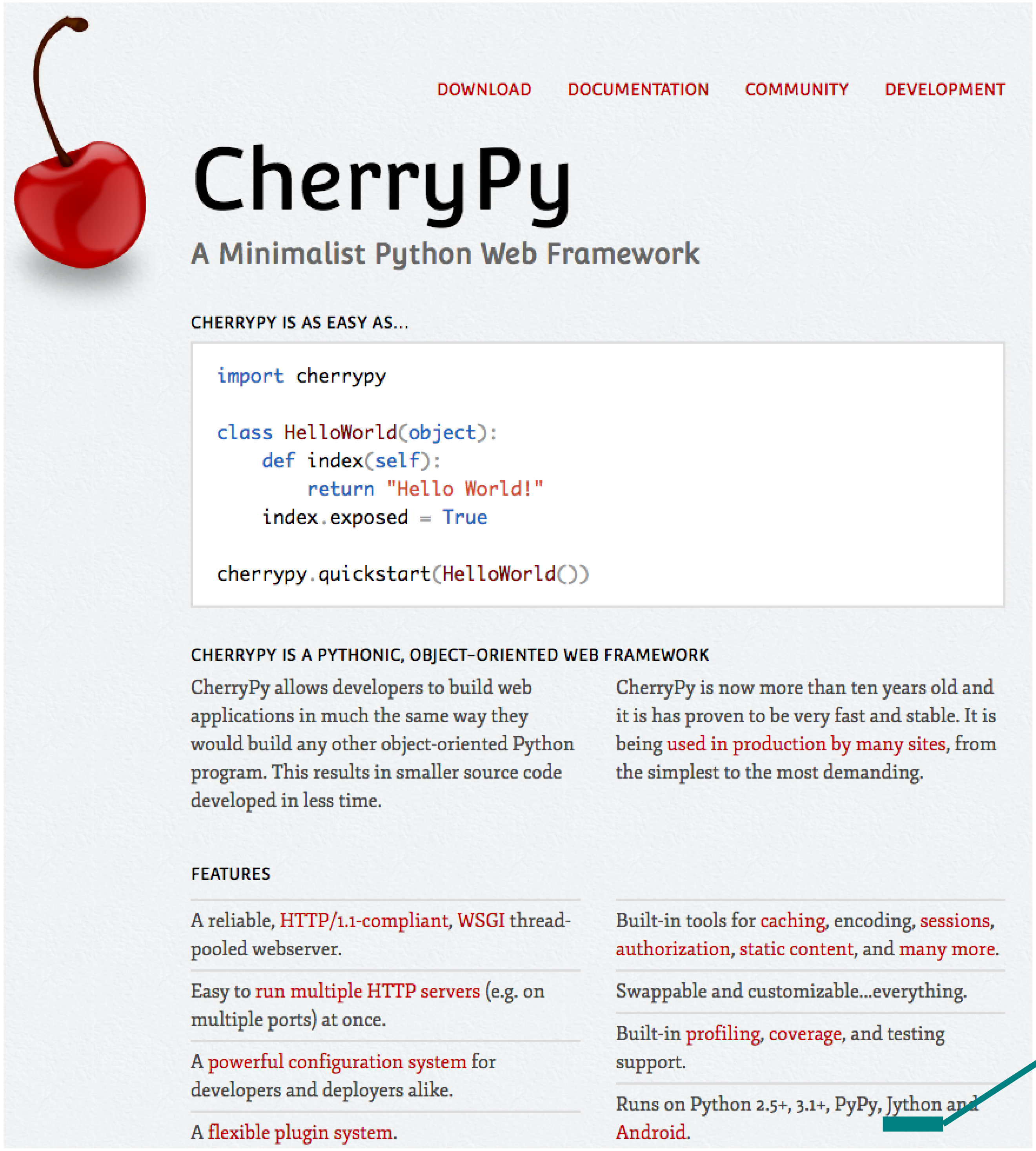
CLAS12 Monitoring Web Service – CherryPy Testing



DOWNLOAD DOCUMENTATION COMMUNITY DEVELOPMENT

CherryPy

A Minimalist Python Web Framework

CHERRYPY IS AS EASY AS...

```
import cherrypy

class HelloWorld(object):
    def index(self):
        return "Hello World!"
    index.exposed = True

cherrypy.quickstart(HelloWorld())
```

CHERRYPY IS A PYTHONIC, OBJECT-ORIENTED WEB FRAMEWORK

CherryPy allows developers to build web applications in much the same way they would build any other object-oriented Python program. This results in smaller source code developed in less time.

CherryPy is now more than ten years old and it is has proven to be very fast and stable. It is being used in production by many sites, from the simplest to the most demanding.

FEATURES

A reliable, HTTP/1.1-compliant, WSGI thread-pooled webserver.

Easy to run multiple HTTP servers (e.g. on multiple ports) at once.

A powerful configuration system for developers and deployers alike.

A flexible plugin system.

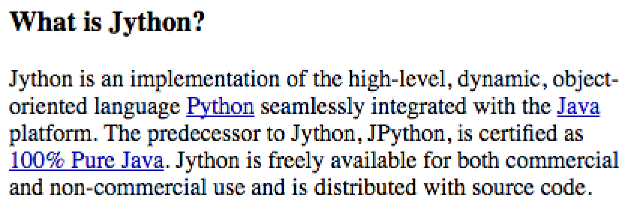
Built-in tools for caching, encoding, sessions, authorization, static content, and many more.

Swappable and customizable...everything.

Built-in profiling, coverage, and testing support.

Runs on Python 2.5+, 3.1+, PyPy, Jython and Android.

* CherryPy was used by CMS



What is Jython?

Jython is an implementation of the high-level, dynamic, object-oriented language Python seamlessly integrated with the Java platform. The predecessor to Jython, JPython, is certified as 100% Pure Java. Jython is freely available for both commercial and non-commercial use and is distributed with source code.

CherryPy, Jython, and Coatjava

- Jython – easy installation, worked out of the box

```
java -jar jython_installer-2.7.0.jar
```

- CherryPy – similar experience

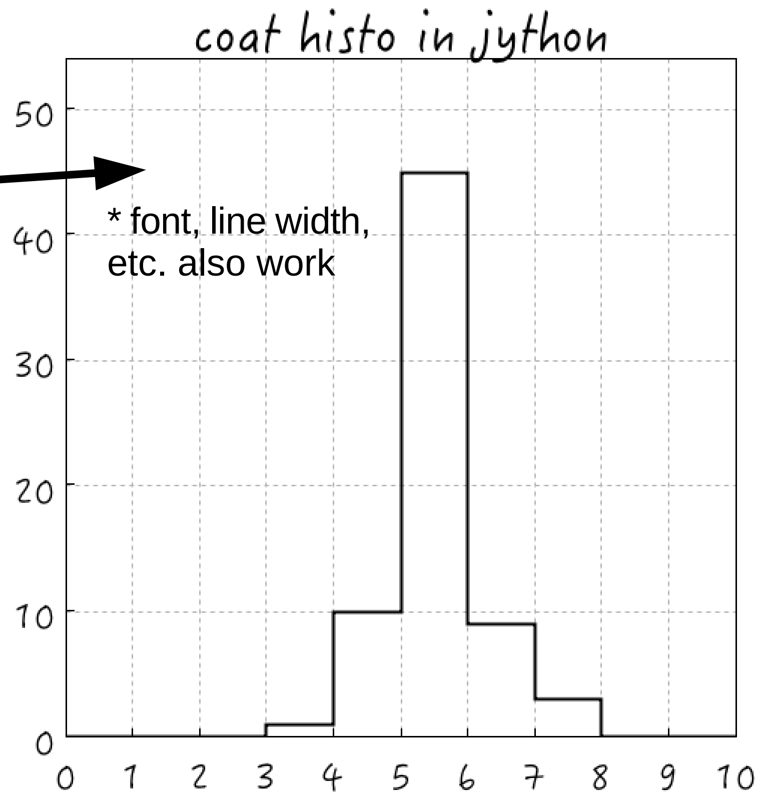
```
$ git clone https://github.com/cherrypy/cherrypy
$ cd cherrypy
$ python setup.py install
```

$ ~/jython2.7.0/bin/jython setup.py install   # one extra line required to run on jython (run with sudo)

- Coatjava and Jython

$ ~/jython2.7.0/bin/jython coatTry.py

```
import sys, os

sys.path.append("/Users/harrison/coatjava-2.4/lib/clas/coat-libs-2.4-SNAPSHOT.jar")

from org.root.histogram import H1D
from org.root.pad import TGCanvas

myHist = H1D("myHist", "coat histo in jython", 10, 0, 10)
myHist.setBinContent(3, 1)
myHist.setBinContent(4, 10)
myHist.setBinContent(5, 45)
myHist.setBinContent(6, 9)
myHist.setBinContent(7, 3)

myCan = TGCanvas("can", "Can", 350, 225, 1, 1)
myCan.draw(myHist)
```



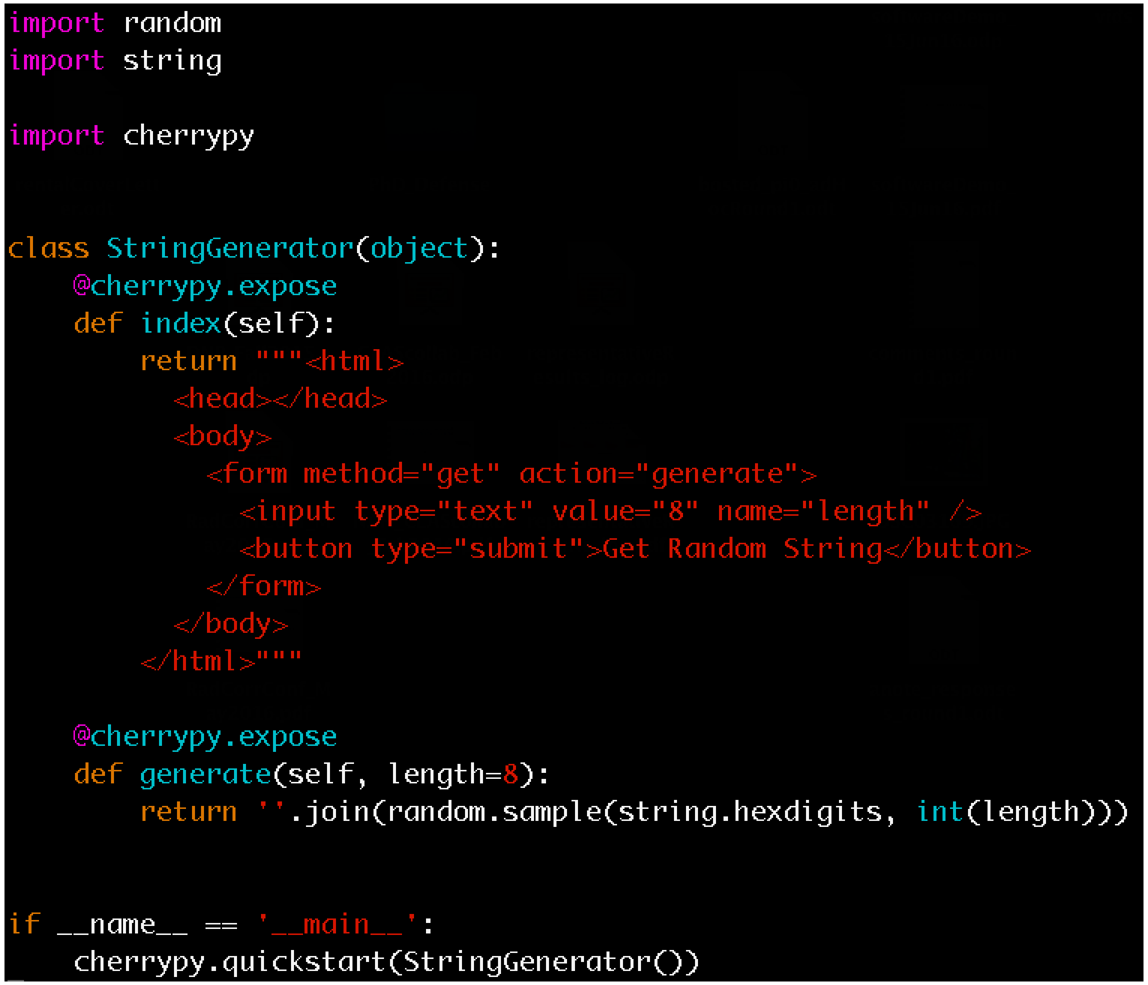
* font, line width, etc. also work

CherryPy, Jython, and Coatjava

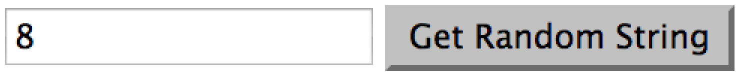
- Jython and CherryPy

Web Browser
http://localhost:8080/

```
$ ~/jython2.7.0/bin/jython tut04.1.py
```



```
import random
import string

import cherrypy


class StringGenerator(object):
    @cherrypy.expose
    def index(self):
        return """<html>
          <head></head>
          <body>
            <form method="get" action="generate">
              <input type="text" value="8" name="length" />
              <button type="submit">Get Random String</button>
            </form>
          </body>
        </html>"""

    @cherrypy.expose
    def generate(self, length=8):
        return ''.join(random.sample(string.hexdigits, int(length)))


if __name__ == '__main__':
    cherrypy.quickstart(StringGenerator())
```



A63405a8

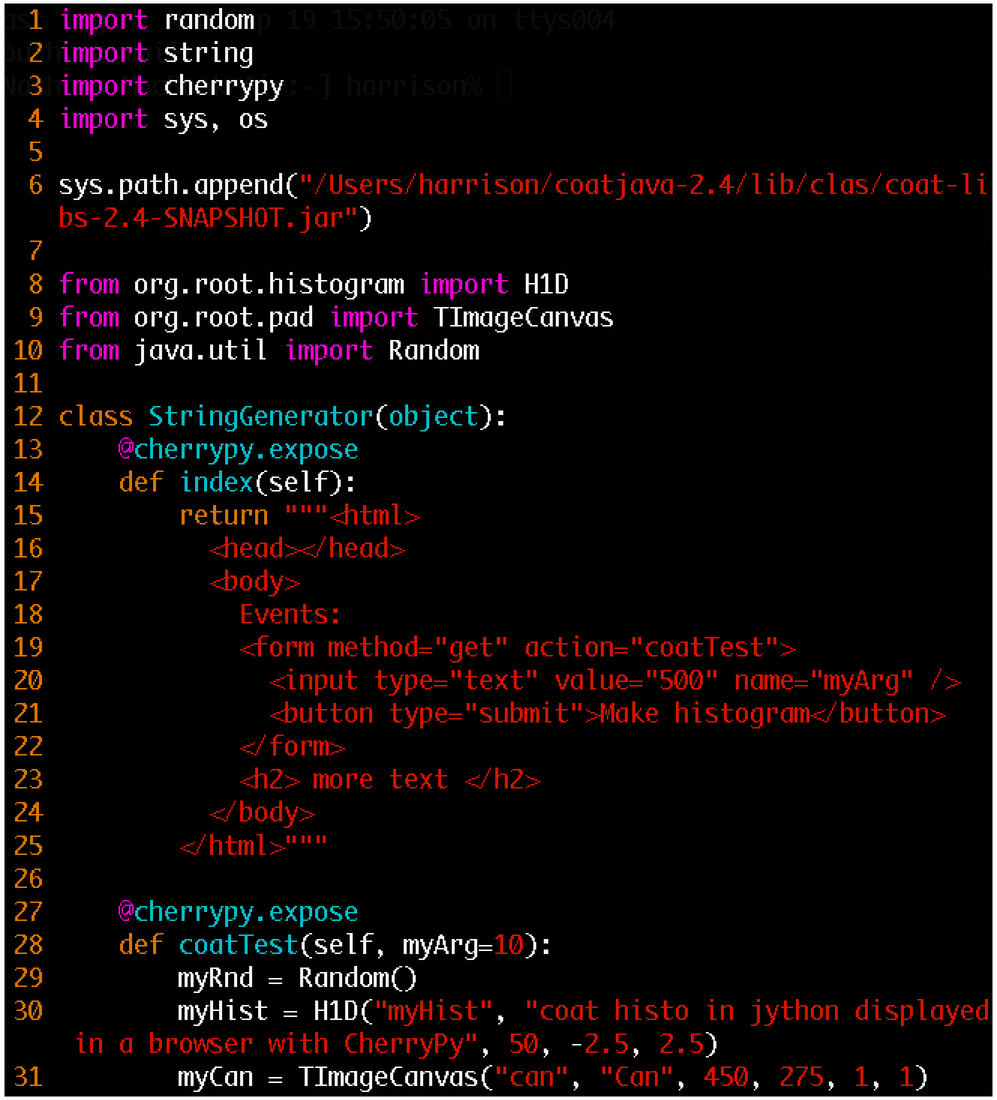
* The following error is produced:
…
Got this failure java.net.ConnectException: Connection refused: /127.0.0.1:8080 during connect (<_realsocket at 0x4 type=client open_count=1 channel=[id: 0xf73cd8b0, 0.0.0.0/0.0.0.0:53383] timeout=0.1>)
…

but it still works...

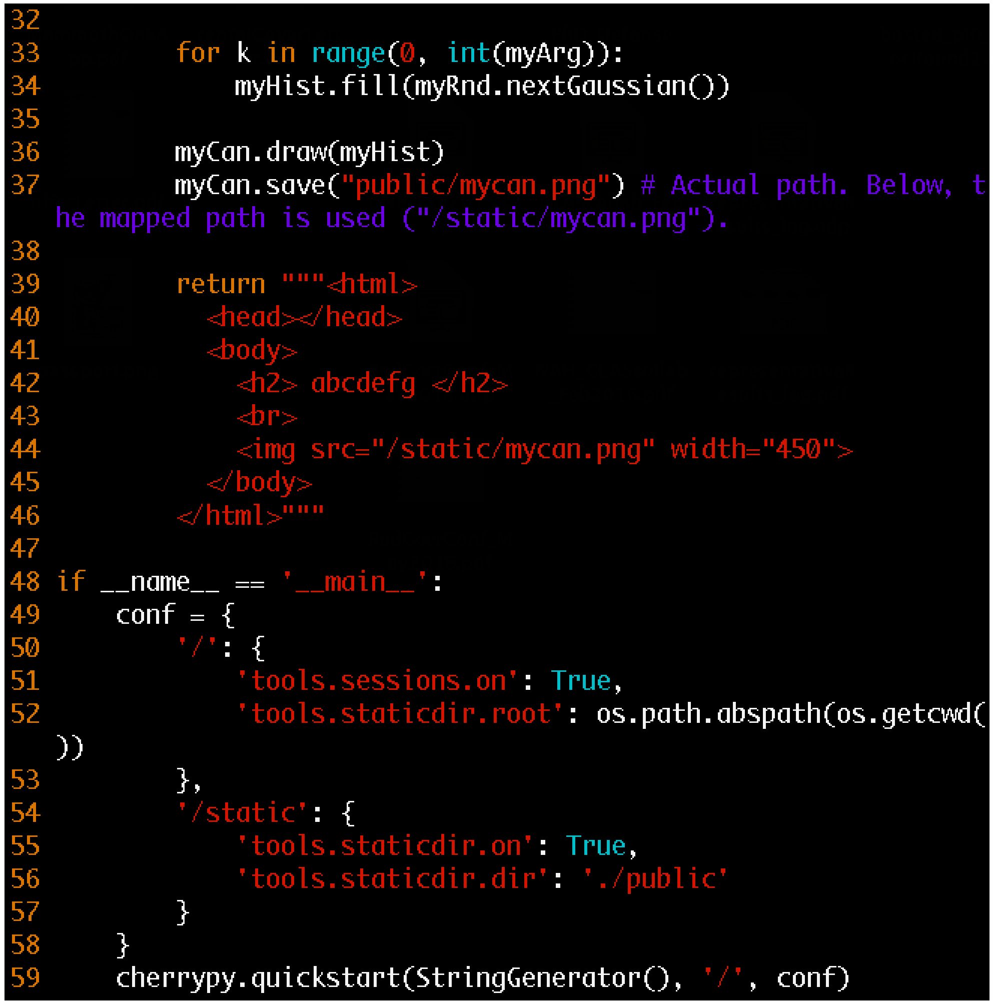
* Do not get this error when using python instead of jython

CherryPy, Jython, and Coatjava

- CherryPy, Jython, and Coatjava



```
import random
import string
import cherrypy
import sys, os

sys.path.append("/Users/harrison/coatjava-2.4/lib/clas/coat-libs-2.4-SNAPSHOT.jar")

from org.root.histogram import H1D
from org.root.pad import TImageCanvas
from java.util import Random

class StringGenerator(object):
    @cherrypy.expose
    def index(self):
        return """<html>
          <head></head>
          <body>
            Events:
            <form method="get" action="coatTest">
              <input type="text" value="500" name="myArg" />
              <button type="submit">Make histogram</button>
            </form>
            <h2> more text </h2>
          </body>
        </html>"""

    @cherrypy.expose
    def coatTest(self, myArg=10):
        myRnd = Random()
        myHist = H1D("myHist", "coat histo in jython displayed in a browser with CherryPy", 50, -2.5, 2.5)
        myCan = TImageCanvas("can", "Can", 450, 275, 1, 1)
```



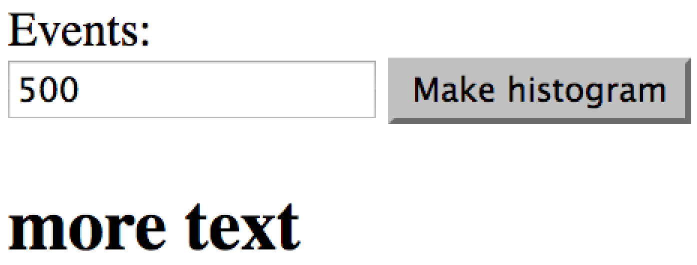
```

        for k in range(0, int(myArg)):
            myHist.fill(myRnd.nextGaussian())

        myCan.draw(myHist)
        myCan.save("public/mycan.png") # Actual path. Below, the mapped path is used ("/static/mycan.png").

        return """<html>
          <head></head>
          <body>
            <h2> abcdefg </h2>
            <br>
            <img src="/static/mycan.png" width="450">
          </body>
        </html>"""

if __name__ == '__main__':
    conf = {
        '/': {
            'tools.sessions.on': True,
            'tools.staticdir.root': os.path.abspath(os.getcwd())
        },
        '/static': {
            'tools.staticdir.on': True,
            'tools.staticdir.dir': './public'
        }
    }
    cherrypy.quickstart(StringGenerator(), '/', conf)
```

```
$ ~/jython2.7.0/bin/jython tut04.2.py
```

CherryPy, Jython, and Coatjava

- CherryPy, Jython, and Coatjava

Web browser
http://localhost:8080/



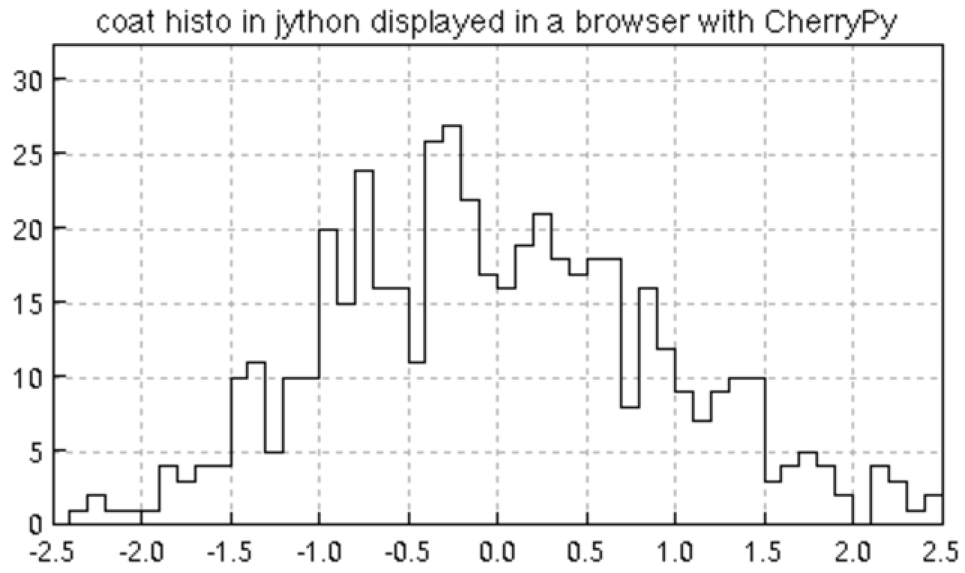
Events:

500

more text

Web browser
http://localhost:8080/coatTest?myArg=500

abcdefg

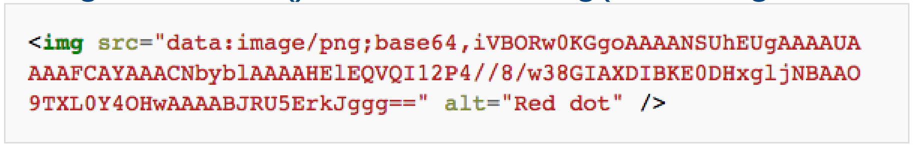


* Simple and inefficient example, but it works! Plenty of room for improvement (e.g. Ajax).

To do:

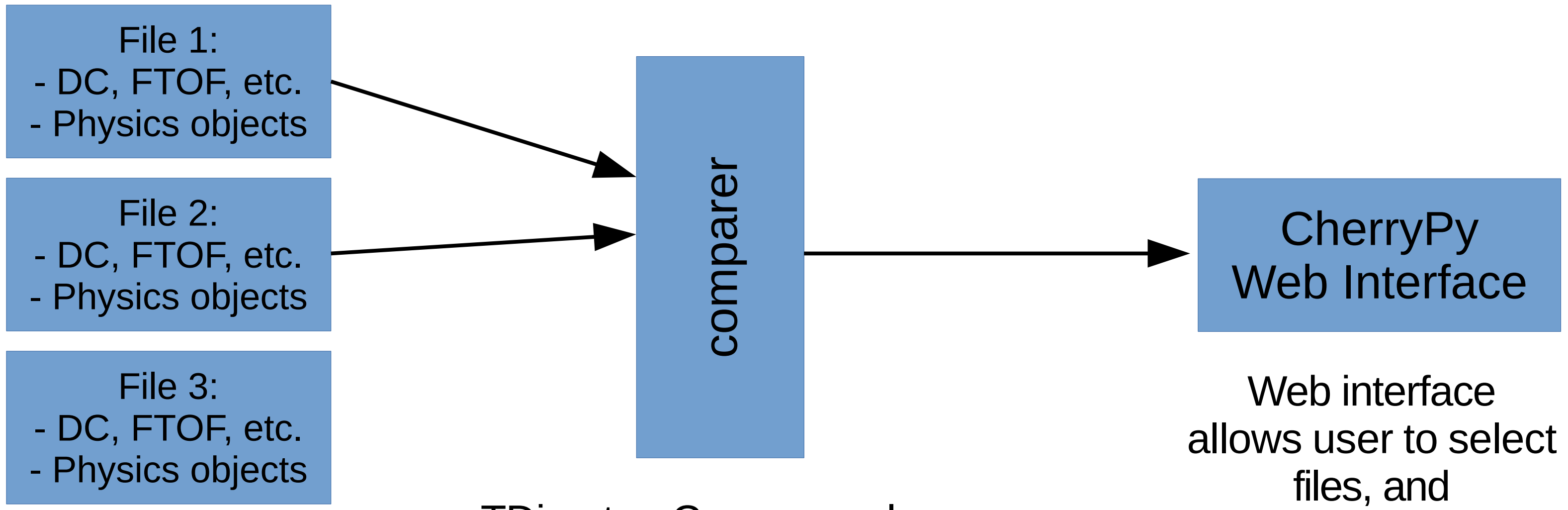
- To display plots, convert the images (Java “BufferedImage”) to a Base64 Image string, then add that string to the HTML, e.g.

```
String str64 =
java.util.Base64.getEncoder().encodeToString(canvas.getCanvasImage());
```



```
<img src="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAUA
AAAFCAYAAACNbyblAAAAHElEQVQI12P4//8/w38GIAXDIBKE0DHxgljNBAAO
9TXL0Y4OHwAAAABJRU5ErkJggg==" alt="Red dot" />
```

- Make an extension of TDirectory class which compares histograms from different files:



Histograms, hipo format

* two sets of files containing histograms for shifters and experts (?)

TDirectoryComparer class extends TDirectory class, does comparisons, statistical tests, etc. Can parse it's structure into HTML collapsable table format.

Web interface allows user to select files, and histograms.

Need a better communication method between the html and the python code.

- Possible solution: “python-requests” (see tutorial 07 on cherryPy webpage)
    - sudo pip install requests
    - for jython:

```
import sys
sys.path.append("/Library/Python/2.7/site-packages")
import requests
```

- Another possible solution: Ajax+jquery (see tutorial 08) see also:

file:///Users/harrison/cherryPy/htmlTable/try3/index.html

/Users/harrison/cherryPy/jython/CLAS12monitor1.py

/Users/harrison/cherryPy/ajEx.py and ajEx.html

for good working examples.