Organization of Configuration

Brett Viren

Physics Department



Daya Bay CN/US Offline, 2009/10/16

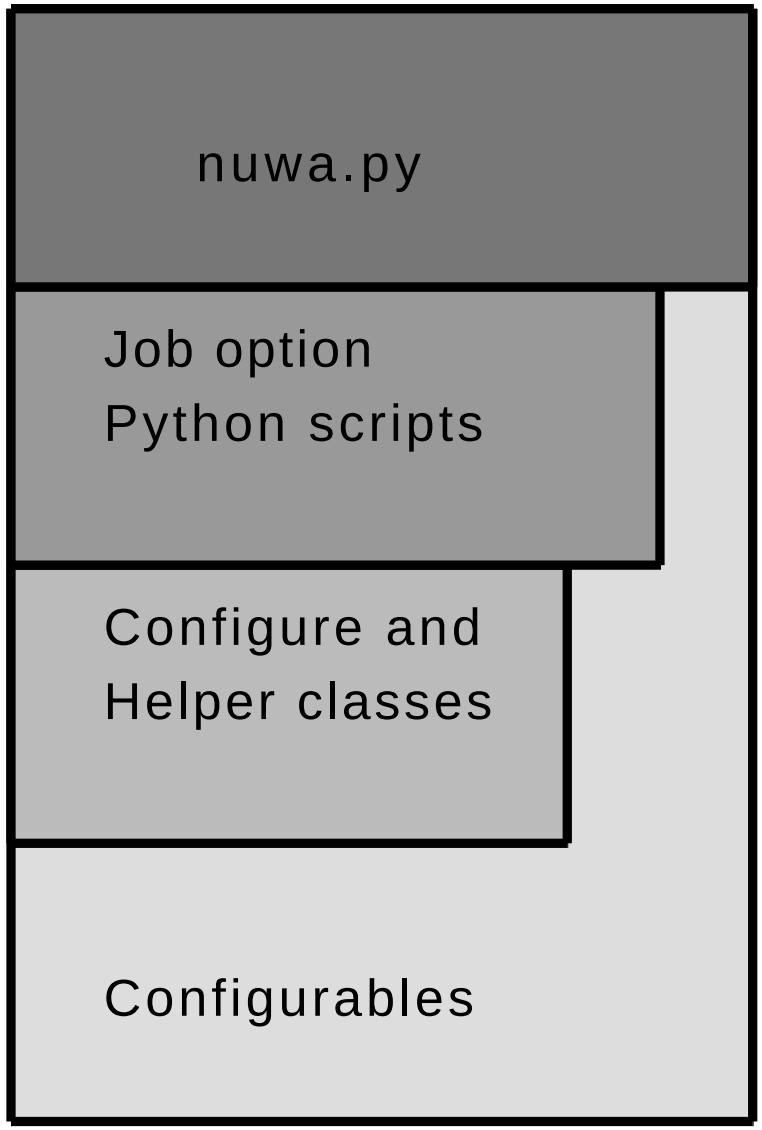
Outline

Layered Configuration Organization

The Layers in Detail

Still To Do

A Layered Approach to Configuration



Gaudi Configurables

- Python classes that contain all Properties of all Components.
- Generated during the build, installed centrally
- Naming pattern: `PackageName.PackageNameConf.ComponentName`

```
# Configurable for ‘‘hello world’’ algorithm
from DybHelloWorld.DybHelloWorldConf import DybHelloWorld
dhw = DybHelloWorld()
dhw.OutputLevel = 3
dhw.MyInt = 42
dhw.MyDouble = 3.14159

# Configure the ApplicationMgr, add to TopAlg list
from Gaudi.Configuration import ApplicationMgr
theApp = ApplicationMgr()
theApp.TopAlg.append(dhw)
```

Configure and Helper classes

- Every package provides a `Configure` class.
- Naming convention: `PackageName.Configure`
- Hand-written to provide minimal, sane and possibly incomplete default configuration for the package's components.
- Supports subsequent customization by user.
- Additional and optional hand-coded "helper" classes

```
from GenTools.Helpers import Gun
gunner = Gun()

import GaudiKernel.SystemOfUnits as units
gunner.timerator.LifeTime = int(60*units.second)
# ...
import GenTools
gt = GenTools.Configure("gun","Particle Gun",helper=gunner)
gt.helper.positioner.Position = [0,0,0]
```

Job option Python files.

- User written and shared
- Bundles "chunks" of **specific** configuration.
  - IBD events generated in far AD #1
  - Electronics with a specific setup
  - Add your personal algorithm(s)
- Job option files should be kept **very** brief. Any generalities should be moved into `PackageName.Configure` or a Helper class.
- Lives in `PackageName/share/*.py`

nuwa.py

- Single, main script everyone uses.
- Installed into user's `PATH`.
- Framework: Python, logging, I/O, geometry, TES/AES.
- Integrates `quanjing.py`.
- Runs Job option Python files.
- Script is minimal, implementation is a class

Current nuwa.py features:

```
shell> nuwa.py

    This is the main program to run NuWa offline jobs.

    It provides a job with a minimal, standard setup.  Non standard
    behavior can made using command line options or providing additional
    configuration in the form of python files.

    Usage:

      nuwa.py [options] [config1.py config2.py ...] [input1.root input2.root ...]

  options (defaults shown):
  -A          |--no-aes               Do not use the Archive Event Store.
  -l 3        |--log-level=3          Set output log level.
  -C no       |--color=no             Use colored logs assuming given background ('light' or '
  -i          |--interactive          Run in interactive mode (def is batch).
  -s False    |--show-includes=False  Show printout of included files.
  -n 0        |--executions=0         Number of times to execute list of top level algorithms.
  -o          |--output=              Output filename
  -O {}       |--output-streams={}    Output file map
  -I {}       |--input-streams={}     Input file map
  -H 8323329  |--hostid=8323329       Force given hostid
  -R 0        |--run=0                Set run number
  -N 0        |--execution=0          Set the starting execution number
  -V          |--visualize            Run in visualize mode
```

Work needed

Still to do, proposed lead in **bold**:

- Convert `ElecSim` and subsequent stages (**Kim**, Brett)
- Convert `RecoEvent` related packages (Xinchun or Liangjian)
- Finish converting `DetSimValidation` (**Brett**, Kevin, Zhe)
- Convert `Fifteen` (Zhe or Brett - TBD)
- General adoption (**all**)

Existing "old-style" scripts should be removed and replaced by brief job option files.