Progress on Configuration Cleanup

Brett Viren

Physics Department

BROOKHAVEN NATIONAL LABORATORY

Daya Bay US Offline, 2009/09/24

Outline

Configurables

Configuration

Going further

Gaudi Bug Fix

- Configurable Tool and Tool Arrays were broken.
- Patch sent to Gaudi, not yet integrated
- Tool owning a Tool is still broken

Configurable Example

```
from Gaudi.Configuration import *
app = ApplicationMgr()
toolsvc = ToolSvc()
from ConfigurableExample.ConfigurableExampleConf import *
a1 = ConcreteAlgorithm("alg1")
pubtool = ConcreteTool("pubtool",Cut=10.01)
toolsvc += pubtool # Pubs must be added to ToolSvc
a1.PublicTool = pubtool
privtool = ConcreteTool("privtool",Cut=20.02)
a1.PrivateTool = privtool
```

Package: `tutorial/ConfigurableExample`.

Current State of Daya Bay Configuration

- Proliferation of main scripts
- Scripts becoming long and complicated
- Necessary changes do not propagate
- Need something better now!

With working configurables can exploit Python modularization.

New Configuration Organization

- Exploit Configurables
- Provide a main script "`nuwa.py`".
- Main script is tiny wrapper to class with the real functionality.
  - Allows other main programs where needed.
- Each package provides helper Python code reduce complexity and provide sane defaults.
  - Convention: main entry point is `PackageName.configure()`.
- Use proper Python modules, classes and functions, not just Python fragments (a'la `athena.py`)

User manual has been updated with more details.

nuwa.py

```
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 ...]

  options (defaults shown):
  -A          |--no-aes               Do not use the Archive Event Store.
  -l 3        |--log-level=3          Set output log level.
  -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
```

```
nuwa.py -n 100 -o ibd.root inverseBeta.py elecTrigReadout.py
```

That `main()` function

```
def main():
    nuwa = NuWa()

    import sys
    nuwa.cmdline(sys.argv[1:])
    if 1 == len(sys.argv):
        nuwa.usage()
        pass

    nuwa.configure()
    return nuwa
```

All the good stuff is in the `NuWa` class.

The `NuWa` class

- Parses arguments into `NuWa.opts`, args become members of this member.
- Configures:
  - Python features (interactivity vs. batch)
  - Random seed algorithm
  - Use of the AES
  - File I/O
- Evaluates any `.py` files given on the command line.
- Any `.root` files will be assumed default input.
- Provides `run()` to start the Gaudi main loop.

Configuring Geometry

```
import XmlDetDesc
XmlDetDesc.configure()

# optional: dump everything, lots of output!
XmlDetDesc.dumper()
```

The `configure()` can also be given a non-default XML starting file.

Configuring GenTools

Too much variety for just one `configure()`, provide suite of examples:

```
import GaudiKernel.SystemOfUnits as units
from GenTools.Examples import Gun, Dumper
gun = Gun(volume = '/dd/Structure/AD/far-oil1',
          particlesPerEvent = 1,
          particleName = "e+",
          momentum = 3*units.MeV)
# Gun saves tools as members for later access:
gun.timerator.LifeTime = 1.0*units.second

# Add Kinematics dumper algorithm
dumper = Dumper()
```

- Provide options to constructor or later through resulting tools stored in data members.
- Need to add other examples (diffuser ball, HepEVT, etc).

Configuring DetSim

```
import DetSim.configure
# only load basic physics
detsim = DetSim.configure()
detsim.physics_list=DetSim.configure.physics_list_basic
detsim.historian(trackSelection="...", vertexSelection="...")
detsim.unobserver(stats=[...])
```

- Constructor takes site & physics list
  - Default it far site & all physics
- Historian & UnObserver via methods
  - Default leaves these not set up

Next steps

- Working with Simon Blyth to get ideas critiqued.
- Now ready for inputs from wider audience.
- Some thought needed for what set of default jobs.
- Need to convert and eradicate all old style scripts.

Comments?....