Table of Contents

Atlas.EventView Tutorial Exercise 1

Exercise 1.a- The Default Atlas.EventView and EVInserterTools

We'll begin learning about EventViews by simply using the standard modules for building EventViews (provided in EventViewBuilerAlgs package) and then just dumping the results to the screen.

There are three default inserter modules

- `DefaultAtlas.EventView` : for reconstructed objects (full simulation)
- `TruthAtlas.EventView` : for Truth objects
- `DefaultAtlfastAtlas.EventView` : for Atlfast objects

The purpose of the inserter modules is the followong:

- Fetches a container of AOD particles from StoreGate
- Iterates over all the particles in the container
- Tests if each particle passes a set of selections defined by job Options (see further down in the job Option file).
- If the particle is selected and does not overlap (delta R match) with any other particle in the final state collection in the Atlas.EventView, "inserts" it to the final state particles.

There are two common operations for inserter modules:

1. change the order of insertion to define priority of addition.
2. change preselection parameters (cuts)

Let's see how these can be achieved using the inserter modules introduced above. To start with, we need a top jobOption file to control the whole job.

There are certain things you need have in the jobOption file. Create a job option file called `EVTutorial_Exercise1.py` in the `run/` directory and start with the following and copy/paste the lines in brown:

```
#################################
# EVTutorial_Exercise1.py
#################################
# files necessary to setup EventView environment
include("EventViewConfiguration/EventViewFullIncludes_jobOptions.py")

# Top sequencer for Athena
from EventViewConfiguration.AlgSequence import *

# instanciate the top alg sequence for athena, only one is required
theJob = AlgSequence()
```

Now we need the `EVToolLooper` to use EVModules, the following does this

```
# import tool looper
from EventViewConfiguration.EVToolLoopers import *

# instanciate one tool looper for reconstructed
defaultEVLooper = EVToolLooper("defaultEVLooper")

# the name of the EventView (in StoreGate) created by this tool looper
defaultEVLooper.EventViewOutputName="defaultEventView"
```

```
# schedule EVToolLoopers to athena top sequencer
theJob += defaultEVLooper
```

What you need to do now is to schedule the inserter module

```
# import the inserter module
from EventViewConfiguration.DefaultEventView_module import *

# define insertion order, you may want to play with this order
toInsert=["Electron", "Photon", "Muon","TauJet","JetTag", "ParticleJet", "MissingEt", "EventInfo"

# schedule the default inserter module with the insertion order as defined above
defaultEVLooper += DefaultEventView("Inserters", toDo=toInsert)

# you can override default preselection parameters, for eg (uncomment if you want to try this)
#defaultEVLooper.Inseerters.ElectronInserter.etCut=30GeV
#defaultEVLooper.Inseerters.JetTagInserter.weightCut=3
```

This will put objects into your Eventview.

You'd want to see what's going on, so lets use ScreenDumper to get output. This will give us a printout out the EventViews in each event.

```
# import screen dumper
from EventViewConfiguration.EVScreenDump_module import *

# schedule screen dumper, printUD=True to see UserData contents
defaultEVLooper+=EVScreenDump("defaultScreenDumper", printUD=False)
```

So this is it for now in terms of scheduling modules. We just need a few final setups.

```
# set up the athena job
theJob.setup()
# print the whole job schedule
print theJob

# obviously, we need input files
#For US ATLAS Standard Model & Higgs Tutorial, use this
#include ("mc11.004100.T1_McAtNLO_top.recon.AOD.v11004101.py")

# For CERN-based Tutorial use this instead
cernSample = [
"rfio:/castor/cern.ch/atlas/csc/valiprod/sampleA/mc11/004201.ZeeJimmy/recon/v11004101/mc11.004201
"rfio:/castor/cern.ch/atlas/csc/valiprod/sampleA/mc11/004201.ZeeJimmy/recon/v11004101/mc11.004201
"rfio:/castor/cern.ch/atlas/csc/valiprod/sampleA/mc11/004201.ZeeJimmy/recon/v11004101/mc11.004201
"rfio:/castor/cern.ch/atlas/csc/valiprod/sampleA/mc11/004201.ZeeJimmy/recon/v11004101/mc11.004201
"rfio:/castor/cern.ch/atlas/csc/valiprod/sampleA/mc11/004201.ZeeJimmy/recon/v11004101/mc11.004201
]
# uncomment to use the cernSample
#EventSelector.InputCollections=cernSample

# specify the number of events you want to run
theApp.EvtMax = 30
```

Now we can try running athena to see the output. From the `run` directory of `Atlas.UserAnalysis`:

```
athena EVTutorial_Exercise1.py > output.log
```

Once the job finishes, examine `output.log`. You'll see a printout of the DefaultAtlas.EventView contents in each event.

Note that you get an output like the following to show the (order of) tools scheduled and prperties set by the modules as well as what you specified.

```
defaultEVLooper schedule ======================================
EVModule/Inserters ( group = ATLAS, author = default )
 |  |-<no properties>
 |- EVElectronInserter/ElectronInserter ( group = ATLAS, author = default )
    |  |-SelectedLabels  : ['ElectronS']
    |  |-OutputLevel     : 5
    |  |-InsertedLabels  : ['Electron']
    |  |-useNN           : False
    |  |-absoluteIsolationCut: 15000.0
    |  |-useIsEM         : True
    |  |-useIsolation    : True
    |  |-onlyEgamma      : True
    |  |-deltaRCut       : 0.1
    |  |-ContainerKey    : ElectronCollection
    |  |-useTRT          : False
    |  |-etCut           : 15000.0
    |  |-nnCut           : 0.7
    |  |-isolationCone   : 0.45
    |  |-OverlapLabels   : ['ElectronOL']
 |- EVPhotonInserter/PhotonInserter ( group = ATLAS, author = default )
    |  |-OutputLevel     : 5
    |  |-SelectedLabels  : ['PhotonS']
    |  |-ContainerKey    : PhotonCollection
    |  |-InsertedLabels  : ['Photon']
    |  |-etCut           : 15000.0
    |  |-OverlapLabels   : ['PhotonOL']
 |- EVMuonInserter/MuonInserter ( group = ATLAS, author = default )
    |  |-SelectedLabels  : ['MuonS']
    |  |-OutputLevel     : 5
    |  |-ContainerKey    : MuidMuonCollection
...
```

Exercise 1.b- Labeling using EVInserterTools

Atlas.EventView allows you to attach string labels to the particles which are stored inside. We'll cover how to add and check particle labels in your code in a later exercise. For now we just note that the `EVInserterTools` allow you to add labels.

The `EVInserterTools` tools allow setting 3 different types of labels:

- Selected
- Inserted
- Overlap

The labeling logic is a follows: for each particle

- Check if the particle is selected
- Check if the particle overlaps with other objects already in Atlas.EventView.
- If overlap and selected: label overlapping particle with Overlap and Selected labels
- If no overlap and selected: insert object and label with Selected and Inserted labels

Seeing the labeling in action makes it clearer. We'll use the default labeling scheme provided for DefaultAtlas.EventView in a module called `DefauleEventViewLabels`. You can easily change how this is done later.

This module uses `SetInserterLabels` python function to save a bit of typing. This function takes an `EVInserterTool` and sets the 3 label properties of the tool. It'll also keep track of all of the various label

names by adding them to the "Labels" list (we'll use this later). Note the job Option configures all of the different inserters such that the X inserter (X=Electron, Photon, etc...) attaches to particles the label "X" on insert, "X"+"S" on selection, and "X"+"OL" on overlap.

Now schedule this module to the tool add the following line to your jobOption **after the line where you added the `DefaultAtlas.EventView` but before `EVScreenDump` so you can see the result**:

```
# import the labeller module
from EventViewConfiguration.DefaultEventViewLabels_module import *
# and schedule it
defaultEVLooper += DefaultEventViewLabels("Labeller", "Inserters")
```

Note that the second argument of this module is the instance name of the inserter module you scheduled since it needs to change the properties of the inserters in this module.

Run again:

```
athena EVTutorial_Exercise1.py > output.log
```

And look at the output file `output.log`. You'll notice that each Final State object now has several labels. For example consider event number 1 (second event):

If you are doing the US-ATLAS SM&Higgs tutorial, you should see this:

```
AthenaEventLoopMgr   INFO   ===>>>  start of event 1  <<<===
defaultEventVie...   INFO ****************************************************
defaultEventVie...   INFO ------------------ EV Screen Dump -------------------
defaultEventVie...   INFO ---------------- Final State Objects ----------------
defaultEventVie...   INFO Object 0:  p_T = 107849 phi = 2.17386 eta = 1.9759
defaultEventVie...   INFO Labels: Electron  ElectronS  JetTagOL  ParticleJetOL  ParticleJetS  Tau
defaultEventVie...   INFO Object 1:  p_T = 17847.9 phi = 0.100586 eta = 0.0655405
defaultEventVie...   INFO Labels: Electron  ElectronS  JetTagOL  ParticleJetOL  ParticleJetS  Tau
defaultEventVie...   INFO Object 2:  p_T = 77090.8 phi = 0.835576 eta = 2.03637
defaultEventVie...   INFO Labels: JetTag  JetTagOL  JetTagS  ParticleJetOL  ParticleJetS
defaultEventVie...   INFO Object 3:  p_T = 71472.4 phi = -1.71487 eta = 2.47688
defaultEventVie...   INFO Labels: ParticleJet  ParticleJetS
defaultEventVie...   INFO Object 4:  p_T = 44762.5 phi = 2.4157 eta = -0.59178
defaultEventVie...   INFO Labels: ParticleJet  ParticleJetS
defaultEventVie...   INFO Object 5:  p_T = 39139.6 phi = -0.257676 eta = -3.31756
defaultEventVie...   INFO Labels: ParticleJet  ParticleJetS
defaultEventVie...   INFO ---------------- Inferred Objects -------------------
defaultEventVie...   INFO ---------------------- UserData ---------------------
defaultEventVie...   INFO eventNumber runNumber MissingEt MissingEtTruth MissingEx MissingExTruth
 MissingEyTruth SumEt SumEtTruth eventWeight
defaultEventVie...   INFO EV does not have Parent
defaultEventVie...   INFO ----------------- End EV Screen Dump ----------------
defaultEventVie...   INFO ****************************************************
AthenaEventLoopMgr   INFO   ===>>>  end of event 1    <<<===
```

If you are running the CERN-based tutorail you would see this

```
AthenaEventLoopMgr   INFO   ===>>>  start of event 1  <<<===
defaultEventVie...   INFO ****************************************************
defaultEventVie...   INFO ------------------ EV Screen Dump -------------------
defaultEventVie...   INFO ---------------- Final State Objects ----------------
defaultEventVie...   INFO Object 0:  p_T = 66023.4 phi = 2.52263 eta = 0.403285
defaultEventVie...   INFO Labels: Electron  ElectronS  JetTagOL  ParticleJetOL  ParticleJetS  Tau
defaultEventVie...   INFO Object 1:  p_T = 24029.8 phi = -1.71924 eta = -0.885241
defaultEventVie...   INFO Labels: Electron  ElectronS  JetTagOL  ParticleJetOL  ParticleJetS  Tau
defaultEventVie...   INFO Object 2:  p_T = 26502.1 phi = -0.84523 eta = -1.10763
defaultEventVie...   INFO Labels: JetTagOL  ParticleJetOL  ParticleJetS  Photon  PhotonS
defaultEventVie...   INFO Object 3:  p_T = 31147.8 phi = 0.467015 eta = -2.03187
```

```
defaultEventVie...   INFO Labels: ParticleJet  ParticleJetS
defaultEventVie...   INFO ---------------- Inferred Objects -------------------
defaultEventVie...   INFO ---------------------- UserData ---------------------
defaultEventVie...   INFO eventNumber runNumber MissingEt MissingEtTruth MissingEx MissingExTruth
defaultEventVie...   INFO EV does not have Parent
defaultEventVie...   INFO ----------------- End EV Screen Dump ----------------
```

From the labels you can tell that the particles are (in order) Electron, Electron, Photon, and ParticleJet. But you also know that particle 0, for example, was reconstructed as a Atlas.JetTag, ParticleJet, and TauJet but only passed the selections for Electron and ParticleJet.

Major updates:
-- AkiraShibata - 09 May 2006

%RESPONSIBLE% AkiraShibata
%REVIEW%

This topic: AtlasArchive > EventViewBuilderTutorialExercise1Conf
Topic revision: r3 - 2007-10-11 - StephenHaywood