XAODMiniTutorialDerivations

Introduction

Welcome to the hands-on session for the derivation framework at the xAOD mini-tutorial! In this session you'll do two things:

- Run existing test derivations and try to understand the output
- Try to implement new derivations according to your physics interests

This session is meant to be an opportunity to experiment with the software with experts available, so please let us know your questions/comments/criticisms etc.

These pages are a step-by-step guide to running the framework - they are not the complete documentation. This can be found here if you want more information.

Running the test derivations

In this first part you'll run the existing test derivations to get a feeling of how things work. These cover eight examples, as follows:

Name	Job options	Description
TEST1	`SkimmingExamples.py`	Simple skimming example
TEST2	`SkimmingExampleStrings.py`	Skimming example using the xAODStringSkimmingTool
TEST3	`StringsToolsExample.py`	Skimming example using the xAODStringSkimmingTool along with an extra variable added by the MCP group
TEST4	`SlimmingExample.py`	Slimming example, using smart slimming
TEST6	`PreSelectionExample.py`	Multi-step filtering example
TEST7	`AugmentationExample.py`	Augmentation and decoration example
TEST8	`ThinningExample.py`	Simple thinning example
TEST10	`TriggerContentExample.py`	Trigger content management

Note that TEST5 and TEST9 are no longer in use.

In this exercise, you'll

- Check out the examples package containing the above formats
- Compile the package using CMake
- Run a set of examples
- If there is time, modify one of the examples, recompile with CMake and then re-run

Later in this tutorial we will have the opportunity to modify the test examples, so to do that we need to have that code locally so we can modify it, compile it, and run it! We will walk you through those steps below The code for these examples lives in a package called DerivationFrameworkExamples which lives in the general athena repository.

Since we are using git now, we need to fork the entire athena repository which holds all the packages, and tell our configuration to only worry about the DerivationFrameworkExamples package. First to fork the athena repository follow the instructions here: Fork the athena repository. You only have to do this once, it's possible you did it in the tutorial earlier this week.

Log into LXPLUS, and make a new directory for this tutorial:

```
ssh -X lxplus.cern.ch
setupATLAS
mkdir DerivationTutorial; cd DerivationTutorial
```

> If you are **not** working on lxplus: you will need to define these variables first:
>
> ```
> export ATLAS_LOCAL_ROOT_BASE=/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase
> alias setupATLAS='source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh'
> setupATLAS
> ```

From our new directory, `DerivationTutorial` let's do the following steps:

```
lsetup git
git atlas init-workdir https://:@gitlab.cern.ch:8443/atlas/athena.git
cd athena/
git checkout -b DerviationFrameworkTests-21.2-20180123 upstream/21.2 --no-track
git atlas addpkg DerivationFrameworkExamples
```

Git should give you some messages to screen that are hopefully successful confirmation of the steps above.

In the above, in the 2nd line we have done a 'sparse' check-out of our forked athena repository. In the 4th line we created a new branch (`git checkout -b`) from which we will work (the name of which is up to you). And finally in the last line we checked out our package of interest, `DerivationFrameworkExamples`.

Now we will set up the latest CMake release of AthDerivation in 21.2 (we will `cd` up one directory first), run CMake to build the makefiles, and then build our local project:

```
cd ../
mkdir build
cd build/
asetup AthDerivation,21.2,latest,slc6
cmake ../athena/Projects/WorkDir/
make
```

Now to run the examples, go up one directory and create a `run/` directory, create a soft link in your directory (`ln -s`) to an xAOD file that we will use as input, source the runtime script, and finally run the job:

```
cd ../
mkdir run
cd run/
source ../build/*/setup.sh
ln -s /afs/cern.ch/atlas/project/PAT/tutorial/cern-july2017/r9315/mc16_13TeV.410501.PowhegPythia8EvtGen_A14_ttbar_hdamp258p75_nonallhad.merge.AOD.e5458_s3126_r9364_r9315/AOD.111
Reco_tf.py --maxEvents 1000 --inputAODFile xAOD.pool.root --outputDAODFile test.pool.root --reductionConf TEST1 TEST2 TEST4
```

- `xAOD.pool.root` is any xAOD file produced with release 21
- Once the job has finished, inspect the log file which is called log.AODtoDAOD. Pay particular attention to the event count and overlaps at the end - when assembling a derivation for a group, you'd be expected to check overlaps with other formats before requesting production.
- Next take a look at the output of the job - there should be four pool.root files corresponding to the four TEST carriages. Some will only contain metadata because the job options don't contain any output stream instructions.
- Some of the carriages can't be run together, in particular TEST6 and TEST7. Can you see why (the error message will help)?

Modifying the package

Now that you have run the framework with the test examples, you should experiment with the software to build a derivation close to your own interests. First, go ahead and browse the currently defined derivations set up by the physics groups. Also you may like to try to run them by substituting TEST1 etc with the name of a derivation. You could also try to set up your own derivations, e.g. by:

- modifying the string-based selections, making selections on a wider range of variables (see the slides for advice on obtaining the variable names)
- adding/removing content via the smart slimming lists (see the slides for advice on obtaining the variable names)
- try decorating the variables or adding extra content via vectors of values
- you could even write a dedicated C++ ISkimmingTool if your selection doesn't easily fit into the string-driven model.
- browse the physics group packages (e.g. the ones not checked out above) for more inspiration
- try downloading a few files from one of the officially produced derivations

The various components that you might consider modifying can be found in the following locations:

- python job options: `$TestArea/../athena/PhysicsAnalysis/DerivationFramework/DerivationFrameworkExamples/share`
- C++ code: `DerivationFrameworkExamples/src`
- C++ headers: `DerivationFrameworkExamples/DerivationFrameworkExamples`

To apply your changes you need to do:

```
cd ../build
cmake ../athena/Projects/WorkDir/
make
cd ../run
source ../build/*/setup.sh
```

You need to do these last five lines every time you want to apply a change.

These instructions are deliberately minimal since the emphasis is on experimenting with the software. Please ask for help if anything is unclear!!! Have fun

Running a real derivation

If you want to create a real derivation that is used by a physics or performance group and you do not want to modify the code, you can produce this derivation using the same `Reco_tf.py` command only using the derivation name after the flag `--reductionConf`. So for example if you want to produce the EGAM1 derivation out-of-the-box, after doing the release setup described above you can do:

```
cd run/
source ../build/*/setup.sh
Reco_tf.py --maxEvents 1000 --inputAODFile xAOD.pool.root --outputDAODFile test.pool.root --reductionConf EGAM1
```

Major updates:
-- JamesCatmore - 10 Jun 2014

Responsible: JamesCatmore
Last reviewed by: **Never reviewed**

Topic revision: r38 - 2019-06-06 - JamesWalder2