Table of Contents

DaVinci Tutorial 0.5 ..... **1**
- Prerequisites and Slides 1
- Creating Ganga template for DaVinci Tutorial jobs, and initial package build 1
- Start to write the options 1
- Configuring DaVinci() 1
- Data 2
- Database tags 3
- Submit 3
- Did it work? 3
- Try Dirac 3
- Understanding the DaVinci Configurable 3

What did I just learn? **4**

Next **5**

DaVinci Tutorial 0.5

We will setup the job and cmtuser area to run the Tutorial jobs in the rest of these tutorials.

Prerequisites and Slides.

- You should have done DaVinciTutorial0 which checks your environment is sound.
- This tutorial corresponds to the slides here: last shown at the October 2012 LHCb software week.

Creating Ganga template for DaVinci Tutorial jobs, and initial package build.

DaVinciTutorial0 introduced you to running jobs inside Ganga, and you set up a generic template. Here we set up a specific one for this tutorial.

1. Start an interactive ganga session as instructed in DaVinciTutorial0
2. At the ganga prompt type:

```
t = JobTemplate( name="DVTutorials", application = DaVinci( version = "v33r7" ))
# t.application.setupProjectOptions = '--nightly lhcb1' # uncomment this if DaVinci version is HE
t.application.getpack( "Tutorial/Analysis v10r6" )
t.application.optsfile = "~/cmtuser/DaVinci_v33r7/Tutorial/Analysis/options/myOptions.py"
t.application.make()
```

Just as `jobs` is a list of previously run ganga jobs, `templates` is a list of previously created instantiations of the `JobTemplate` class. This template can be used for all the Tutorial analyses, and if you ever restart ganga you can just copy it off the `templates` list.

Start to write the options

1. First make sure you have the LHCb flavour of emacs. If not do what's described here.
2. Open the main options file (from ganga type:)

```
!emacs $t.application.optsfile[0].name &
```

1. Write in the options file.

```
from Gaudi.Configuration import *
from Configurables import DaVinci
DaVinci()
```

Configuring `DaVinci()`

We now need to configure the job in the main options file. In this case it is as simple as writing some options for the "DaVinci Configurable".

Remember from the tutorial that this object is **not** the same as the Ganga GPI object called "DaVinci". You need to add these lines to your **options file**, not paste them into the command line, they are used to steer the actual Gaudi, not define some parameter of the ganga job.

```
DaVinci().HistogramFile = "DVHistos.root" # Histogram file
DaVinci().EvtMax = 1000 # Number of events
DaVinci().DataType = "2012" # Default anyway
DaVinci().TupleFile = "DVnTuples.root" # A file to store nTuples, such as the Lumi ntuple
DaVinci().Lumi = True # integrate the luminosity FSRs and print/store the result
```

Hint: if you have closed the file accidentally you can reopen with:

```
!emacs $t.application.optsfile[0].name &
```

Data

Then we need some data.

- We will attach the data this time directly to the job object, instead of writing it into an options file. This is much more flexible and almost always works first time.
- If you have a very good connection, you can open the `feicim` browser in ganga.

```
data = browseBK()
```

- - ♦ Navigate to a data sample of 2012 data -> reco 14 -> stripping 20 -> Dimuon stream.
  - ♦ Double click on it, and "save". Check:

```
for i in data.files: print i
```

- If you do not have a very good connection,, you can use "BKQuery" inside Ganga to get the data.
  - ♦ Try to get the bookkeeping path from the Bookkeeping webpage,
  - ♦ the path will appear on the top between the big "+" and "<-Go", just copy it (and remove things like "(Full stream)"),
  - ♦ then get the data in Ganga, like

```
data=BKQuery('/LHCb/Collision12/Beam4000GeV-VeloClosed-MagUp/Real Data/Reco14/Stripping20/9000000
```

- This will be a set of LFNs that you can pass to the job:

```
t.inputdata = data[0:10] # only access the first 10 elements for speed as this is a tutorial
```

Remember, datasets can be stored in:

- '.py' Files, using the lhcb_bkk interface outside of Ganga. This is not really recommended.
- '.py' Files by using the 'export' feature inside Ganga. This is not really recommended.
- The ganga jobs themselves, as `LHCbDatasets`. This works well, until you delete the jobs, and means you would have to give your jobs clever names.
- The templates, as `LHCbDatasets`. This also works well,
- The ganga box, the recommended and flexible way. Here you can give datasets different names, and even add "BKQuery" objects `box.add(j.inputdata, "R14S20 DIMUON")` ... any ganga object can be stored in your box.

Database tags

This part is rather complicated.

- LHCb data is reconstructed assuming a given detector geometry and alignment.
- These are stored in a database, which is updated regularly with new alignment conditions
- It is recommended, though, that you **always** use the same tags as were used to create the file (unless there was some known bug)
- Currently the default DaVinci tags for 2012 data will work for you, these should be:

```
DaVinci().DDDBtag="dddb-20120831"
DaVinci().CondDBtag="cond-20120831"
```

- You can find the correct database tags from the book-keeping, by clicking "more information" when you get to the leaf with the files actually in it.
- If you use directly the dirac lhcb_bkk command (in a different window) it will save a file in which the database tags should be listed.
- Otherwise, ask a colleague working on similar data.

Submit

```
j = Job(t)
j.submit()
```

Did it work?

- You tell me!
- Take a look at the standard out and the XMLSummary ... can you understand it?

Try Dirac

- If you're brave, and if the local test worked, try submitting the job to the grid!

```
j=j.copy()
j.backend=Dirac()
j.submit()
```

Understanding the DaVinci Configurable

- "DaVinci" means several things in our software. Really the meaty part of "DaVinci" you will interact with most is the python configuation object written into the options files.
- This is not the same object as you get by typing `DaVinci()` at the ganga prompt! That is just a Ganga job object, known as a GPI-object, it doesn't really do anything apart from telling ganga what type of environment to set up.
- This configurable `DaVinci()` in your options files is very different
- This simple-looking python object is actually very complicated and has many possible options, all listed in For more details see: Doxygen.
- For you it is meant to simplify your life, so mostly the default options work, and mostly you will simply be configuring existing algorithms to write out some sort of nTuple for your analysis.

.

What did I just learn?

- You performed getpack inside of Ganga to configure a job ready to run the rest of the tutorials.
- You also had a first stab at the book-keeping, and learnt a little about database tags
- You now know a little about the DaVinci configurable, and how to set it up correctly. For more details see: Doxygen DaVinci Configurable

Next

- If you want to learn how to read a Stripped DST and do some cool stuff with it, move to DaVinciTutorial8.
- If you want to learn how to use existing options files, and existing C++, move to DaVinciTutorial4.
- If you want to learn how to write your own options files, and your own C++, move to DaVinciTutorial1.

--- This topic: LHCb > DaVinciTutorial0p5
Topic revision: r34 - 2015-10-14 - CayoMarCostaSobral