Table of Contents

DaVinci Tutorial 0

We start with just running DaVinci for the first time, and getting some meaningful output. If you ever need to go back to the drawing board, you can start with this tutorial.

Read this first

This tutorial will not work with the Grid as of 2014-08-17 (probably due to changes in the Dirac API). It will still work locally, just don't be surprised if you try it with the option

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

and it doesn't work.

Prerequisites and Slides.

- Ideally you should really have done the GangaTutorial1, but it is not absolutely necessary.
- This tutorial corresponds to the slides here: last shown at the October 2012 LHCb software week.

Creating Ganga template for DaVinci jobs

Ganga is the preferred frontend for LHCb jobs, before you start with DaVinci you should really have done the GangaTutorial1, but it is not absolutely necessary.

To run whichever DaVinci you want, you don't and/or shouldn't need to store anything locally. On most CERN computers and on a lot of institutes the CERN software is already installed for you. At most the only thing you need locally should ideally be an options file. If your options are stored in Erasmus, you might not even need that!

You might already have done part of this. It doesn't harm to redo it.

1. Set environment for Ganga:

```
SetupProject Ganga
```

- ♦ and take the latest version (5-8-16 at time of writing).
- ♦ If you have last used `ganga` an age ago, or this is your first time

```
ganga -g
```

1. Now start an interactive session:

```
ganga
```

1. At the ganga prompt type:

```
t = JobTemplate( name="DVTutorial0", application = DaVinci())
```

1. You can access the environment for the package directly in Ganga, and setup a simple options file taken from the standard DaVinci options.

```
envDict=t.application.getenv() #example to demonstrate what ganga does behind-the-scenes (takes a
print envDict # the result of calling SetupProject DaVinci
envDict['DAVINCIROOT']
```

```
dir=envDict['DAVINCIROOT'] + '/options'
ls $dir
t.application.optsfile = dir+'/DaVinci-Simplest.py'
```

NB: Be very careful not to copy-paste leading spaces; python does not like that at all.

Running it!

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

Once the job has completed, you can peek the stdout, and get familiar with what DaVinci looks like for such a simple job.

Submit Options

If you want the output to a file instead of to the terminal do:

```
j.backend=Local()
```

Now you have a job which works, you can try submitting the same job to `LSF` or even `Dirac` backends.

The input file used for this tutorial, though, is a PFN located at CERN (for simplicity)... in the next step we will get some data from the book-keeping, allowing you to run anywhere in the world via the GRID.

What did I just do?

- You made a "job" inside "Ganga".
- A Ganga job is a set of instructions to run something, somewhere, and is very flexible.
- This job was told to configure DaVinci and then call `gaudirun.py` with your options file
- Ganga objects have their own help, can be printed and/or exported to save for later
- Ganga objects are stored for you in your Jobs (automatic) Templates (automatic) and Box (manual)

```
print j
jobs #hit enter
templates #hit enter
box #hit enter
help(box)
help(j.application)
export(j , "Tutorial0.py")
!emacs Tutorial0.py &
```

Understanding the XMLSummary

You'll need to use the latest version of ganga (5-8-16) to ensure this part works.

The stdout is very confusing, however it can be condensed into an xml file for easy reading.

- This XMLSummary can be added to any LHCb application, and it is **Automatically added** by the latest Ganga.
- peek the summary.xml file, which will list the files you have interacted with, the counters you have produced in algorithms, and the status of the job.
- `j.peek("summary.xml")`

- The XMLSummary can be interpreted directly in python, and is documented with a full FAQ here:XMLSummaryFAQ
- The XMLSummary can be merged using a standard ganga merger (unlike stdout which is tough to merge)
- Also there are some nice new ways of interpreting this information directly inside Ganga with the j.metadata.

Understanding the options

1. Open up the options file you just submitted...(from ganga type:)

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

1. Can you understand what it is doing?
2. Compare this with what appears in the XMLSummary and the stdout.
3. Does the job do what you expect?
4. Was the job successful?

`DaVinci-Simple.py` is the simplest options file required to run DaVinci and make some output you can play with in other tutorials. Notice that there are still several parts you need to set yourself even in the simplest case. Ask a colleague what they are for if you are stuck.

.

What did you learn?

- You now have a DaVinci job which runs and produces some output.
- This is the first hurdle in LHCb software, congratulations!!
- Now that you have one job which works, you only need to copy and edit it around with `j.copy(True)`, or by editing the template you made above, to produce anything you might want ...
- You can return to this tutorial whenever you have a job which doesn't work, to check your environment is not screwed up.

Next

- Tutorial 0.5 will interface this now working job with the tutorial packages needed later (many examples and cool things to try)
- Move onto the Tutorial in DaVinciTutorial0p5

--- This topic: LHCb > DaVinciTutorial0
Topic revision: r37 - 2014-06-17 - NathanaelFarley