DaVinci Configurable

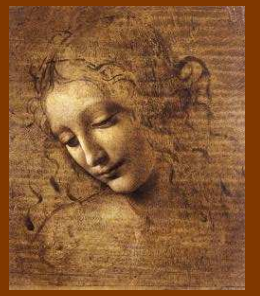
- The `DaVinci()` configurable
- `DaVinci()` for newbies
  - What is does
  - I/O
- `DaVinci()` for experienced users
- To do list

#2/6

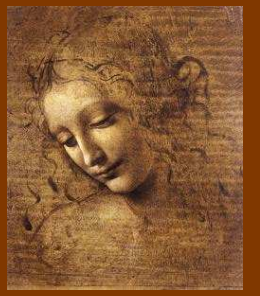
March 2009 Tutorials

Patrick Koppenburg
Imperial College
London

Main change



- `DaVinciCommon.opts` is gone.
  - ➜ Any file including it will not work.
  - ➜ Most preselections are broken. But magically the stripping still works.
    - We'll see later how to get your selection to work.
- All functionality moved into the `DaVinci()` configurable, and other configurables in subpackages.
  - ➜ Is needed to allow running on 2008 and DC06 data without having to remember the tiny differences.
  - ➜ Gives **DaVinci** the same look and feel as **Brunel**.
    - Guarantees a correct ordering of algorithms as everything is controlled by the configuration.
      - ✗ I'm sure some use-cases have not been foreseen and will need fine-tuning
    - ✔ I hope we'll be able to keep the interface stable for some time.



DaVinci() for newbies

Typical DaVinci main file

```
from Gaudi.Configuration import *
from Configurables import GaudiSequencer
##############################################################
importOptions("$STDOPTS/PreloadUnits.opts")
importOptions("$B2DILEPTONROOT/options/DoPreselBu2LLK.opts")
preselSeq = GaudiSequencer("SeqPreselBu2LLK")
##############################################################
from Configurables import PrintTree, PhysDesktop
tree = PrintTree("PrintBu2LLK")
exampleSeq.Members += [ tree ]
tree.addTool( PhysDesktop )
tree.PhysDesktop.InputLocations = [ "PreselBu2LLK" ]
##############################################################
# Standard configuration
from Configurables import DaVinci
DaVinci().EvtMax = 1000                          # Number of events
DaVinci().SkipEvents = 0                         # Events to skip
DaVinci().DataType = "2008"                      # Default is "DC06"
DaVinci().Simulation   = True
DaVinci().HistogramFile = "DVHistos_1.root"      # Histogram file
DaVinci().TupleFile = "DVNtuples.root"           # Ntuple
DaVinci().UserAlgorithms = [ preselSeq ]         # User algorithms
DaVinci().MoniSequence = [ tree ]                # Monitoring
# DaVinci().MainOptions  = "" # None
```

What does it do?

- Standard **Gaudi/LHCb** initialisation
- Data on demand service
  - Unpacking
  - Common particles . . .
  - ➜ Depends on data type
- Init sequence
  - Redo protoparticle PID
  - Redo MC links
  - ➜ Depends on data type
- Sequencing of algorithms
  - Init
  - L0
  - HLT
  - User algorithms
  - Monitoring
- I/O handling
  - Input, events, skip . . .
  - Histogram file
  - Ntuple file
  - DST, ETC

Ordering

DaVinci will make sure the algorithms are run in this order:

1. Input sequence (calibrations, etc)
2. L0 (if asked for or enforced by HLT)
3. HLT (if asked for)
4. User algorithms
5. User Main options
6. Monitoring algorithms

DC06: Re-run muon ID (slow)

```
EVENT LOOP                          |    89.096 |   153.765 |   23
 DaVinciInitSeq                     |    54.782 |    94.262 |   12
  DaVinciInit                       |     0.170 |     0.177 |    0
  PhysInitSeq                       |    54.612 |    94.070 |   12
   TestProtoPRecalibration          |    54.612 |    94.066 |   12
    CheckProto                      |     0.450 |     1.283 |    0
    ProtoPRecalibration             |    54.142 |    92.777 |   12
     MuonRec                        |     2.150 |    12.125 |    2
     MuonID                         |    47.793 |    63.846 |    4
     UpdateMuonPIDInProtoP          |     3.429 |    13.054 |    1
     ChargedProtoCombineDLLsAl      |     0.410 |     0.429 |    0

     RichPIDsFromProtos             |     0.320 |     3.291 |    0
  AnalysisInitSeq                   |     0.000 |     0.001 |    0
 DaVinciMainSequence                |     8.129 |    32.725 |    3
  SeqPreselBu2LLK                   |     8.009 |    32.606 |    3
   DiLeptonForPreselBu2LLK          |     0.180 |     0.206 |    0
   PreselBu2LLK                     |     0.471 |     0.582 |    0
   PrintPreselBu2LLK                |     0.000 |     0.000 |    0
  BTagging                          |     0.000 |     0.002 |    0
 MonitoringSequence                 |     1.090 |     1.131 |    0
  ExampleSeq                        |     1.090 |     1.128 |    0
   PrintBu2LLK                      |     1.050 |     1.051 |    0
*   StdDC06LooseMuonsSeq            |     0.360 |     0.342 |    0
*    StdDC06LooseMuons              |     0.050 |     0.056 |    0
*   StdDC06LooseElectronsSeq        |     2.520 |    12.442 |    0
*    StdDC06LooseElectrons          |     0.080 |     0.052 |    0
*   StdDC06LooseKaonsSeq            |     0.235 |     0.321 |    0
```

Ordering

DaVinci will make sure the algorithms are run in this order:

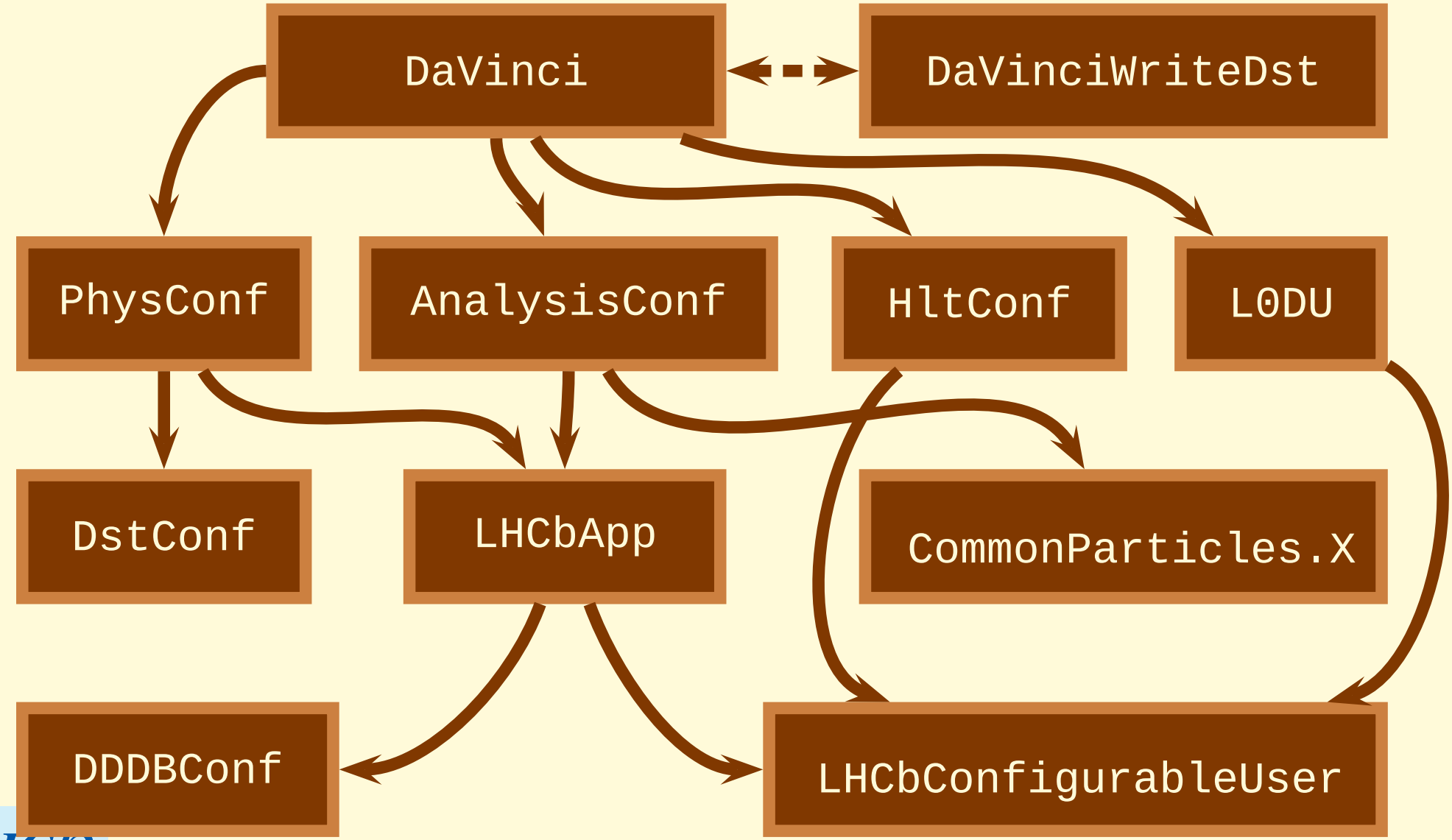
1. Input sequence (calibrations, etc)
2. L0 (if asked for or enforced by HLT)
3. HLT (if asked for)
4. User algorithms
5. User Main options
6. Monitoring algorithms

DC06: Re-run muon ID (slow)

2008: Get muon ID from protos

```
EVENT LOOP                        |    12.038 |    19.066 |
 DaVinciInitSeq                   |     3.929 |     9.227 |
  DaVinciInit                     |     0.100 |     0.141 |
  PhysInitSeq                     |     3.799 |     9.079 |
   TestProtoPRecalibration        |     3.799 |     9.076 |
    CheckProto                    |     0.190 |     0.286 |
    ProtoPRecalibration           |     3.599 |     8.786 |
     MuonPIDsFromProtos           |     3.489 |     8.705 |
     RichPIDsFromProtos           |     0.090 |     0.073 |
  AnalysisInitSeq                 |     0.000 |     0.002 |
 DaVinciMainSequence              |     6.839 |     8.542 |
  SeqPreselBu2LLK                 |     6.749 |     8.450 |
   DiLeptonForPreselBu2LLK        |     0.200 |     0.141 |
   PreselBu2LLK                   |     0.174 |     0.166 |
   PrintPreselBu2LLK              |     0.000 |     0.000 |
  BTagging                        |     0.000 |     0.002 |
 MonitoringSequence               |     0.940 |     0.962 |
  ExampleSeq                      |     0.930 |     0.960 |
   PrintBu2LLK                    |     0.880 |     0.890 |
*   StdDC06LooseMuonsSeq         |     0.160 |     0.170 |
*    StdDC06LooseMuons            |     0.050 |     0.041 |
*   StdDC06LooseElectronsSeq      |     0.870 |     1.954 |
*    StdDC06LooseElectrons        |     0.010 |     0.038 |
*   StdDC06LooseKaonsSeq          |     0.152 |     0.158 |
```

Structure


DaVinci
DaVinciWriteDst
PhysConf
AnalysisConf
HltConf
L0DU
DstConf
LHCbApp
CommonParticles.X
DDDBConf
LHCbConfigurableUser

DaVinci() options

```
# Application Configuration : sent to LHCbApp and Gaudi
DaVinci().EvtMax               :  -1        # Number of events to analyse
DaVinci().SkipEvents           :   0        # Number of events to skip at beginning for file
DaVinci().PrintFreq            : 100        # The frequency at which to print event numbers
DaVinci().DataType             : 'DC06'     # Data type, can be ['DC06','2008'] Forwarded to PhysConf
DaVinci().Simulation           : True       # set to True to use SimCond. Forwarded to PhysConf
DaVinci().DDDBtag              : 'default'  # Tag for DDDB. Default as set in DDDBConf for DataType
DaVinci().CondDBtag            : 'default'  # Tag for CondDB. Default as set in DDDBConf for DataType
# Input
DaVinci().Input                : []         # Input data. Can also be passed as a second option file.
# Input/Output
DaVinci().HistogramFile        : ''         # Write name of output Histogram file
DaVinci().TupleFile            : ''         # Write name of output Tuple file
DaVinci().ETCFile              : ''         # Name of ETC file
DaVinci().InputType            : 'DST'     # or 'DIGI' or 'ETC' or 'RDST' or 'DST'. Nothing means the input typ
# DaVinci Options
DaVinci().MainOptions          : ''         # Main option file to execute
DaVinci().UserAlgorithms       : []         # User algorithms to run.
# Monitoring
DaVinci().MoniSequence         : []         # Add your monitors here
DaVinci().RedoMCLinks          : False      # On some stripped DST one needs to redo the Track<->MC link table.
# Trigger running
DaVinci().L0                   : False     # Run L0.
DaVinci().ReplaceL0BanksWithEmulated : False # Re-run L0
DaVinci().HltType              : ''         # HltType : No Hlt. Use Hlt1+Hlt2 to run Hlt
DaVinci().HltUserAlgorithms    : [ ]        # put here user algorithms to add
DaVinci().Hlt2Requires         : 'L0+Hlt1' # Say what Hlt2 requires
```

The various `DaVinci()` sequencers

```
DaVinci().UserAlgorithms      : []    # User algorithms to run.
DaVinci().MainOptions         : ''    # Main option file to execute
DaVinci().MoniSequence        : []    # Add your monitors here
```

- There's no fundamental difference
- They will be executed in that order
- I like to put my selection in `UserAlgorithms` and my Tuple-filling in `MoniSequence`.

Trigger

```
DaVinci().HltType = "Hlt1+Hlt2"                    # Both Hlt levels
DaVinci().Hlt2Requires = 'L0+Hlt1'                 # Remove Hlt1 to Ignore it in
DaVinci().L0 = False                               # Re-Run L0
DaVinci().ReplaceL0BanksWithEmulated = False       # Redo L0 banks
```

- `HltType` defines which levels to run. Can be `'Hlt1'`, `'Hlt2'` or `'Hlt1+Hlt2'`.
- `Hlt2Requires='L0+Hlt1'` (default) runs Hlt2 only if Hlt1 passed. Can be changed to `'L0'` or `''`.
- You can then recover the Hlt1 efficiency for Hlt2 accepted events in `TupleToolTrigger`.

➜ That's how I generate the Hlt stats page at

`https://twiki.cern.ch/twiki/bin/view/LHCb/Hlt2EffsRates`

Trigger

```
DaVinci().HltType = "Hlt1+Hlt2"                    # Both Hlt levels
DaVinci().Hlt2Requires = 'L0+Hlt1'                 # Remove Hlt1 to Ignore it in
DaVinci().L0 = False                               # Re-Run L0
DaVinci().ReplaceL0BanksWithEmulated = False       # Redo L0 banks
```

- `L0=True` re-runs L0 using the L0-banks available on the input data. It allows to use a new tuning if available, but not new algorithms.
- `ReplaceL0BanksWithEmulated` runs the L0 emulator and replaces what is in the input data. It enforces `L0=True`.
- This is enforced on DC06 data :

```
 seqL0                          |    15.847 |    16.708 |   10.086      90.1 |
  L0DUBankSwap                  |    15.844 |    16.704 |   10.082      90.1 |
   RemoveL0Banks                |     0.030 |     0.036 |    0.028       0.5 |
   ReRunL0                      |    15.808 |    16.658 |   10.048      89.6 |
    L0Calo                      |     0.817 |     0.911 |    0.473      18.6 |
    L0Muon                      |    13.328 |    13.952 |    7.968      76.7 |
    L0PuVeto                    |     0.301 |     0.325 |    0.166       5.4 |
    L0DU                        |     1.327 |     1.452 |    1.241      58.1 |
```

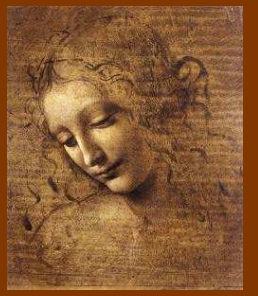
DaVinciTestData.py

- Provides test data as before
- But provides different data depending on `DataType` option

  DC06: stripped bb

  2008: inclusive J/ψ
- To test your options do:

```
gaudirun.py options/MyOpts.py options/DaVinciTestData.py
```



I/O

Histograms and Tuples

`DaVinci()` hides the calls to `HistogramSvc` and `NTupleSvc` from the user:

```
signal = 'Bs2DsDs'
DaVinci().TupleFile =  "HLT-"+signal+".root"
DaVinci().HistogramFile = "DVHlt2-"+signal+".root"
DaVinci().ETCFile = "MyETC.root"
```

`TupleFile` will contain all tuples. If empty, nothing is written (default).

`HistogramFile` will contain all histograms. If empty, nothing is written (default).

`ETCFile` will contain all Event tag collections. If empty, nothing is written (default).

➜ This assumes some algorithms do produce histos, Tuples & ETCs. . .

These are root files. Hbook is not supported (what's hbook?)

What is an ETC ?

- It's a `TTree`
- Which contains a pointer to an event in a file
- so it can be used to access an event. For instance:

```
DaVinci().Input   = [
    "COLLECTION='TagCreator/EventTuple' DATAFILE='DVPresel_ETC2.root'
     TYP='POOL_ROOT' SEL='(PreselBd2KstarMuMu>0)'" ]
```

- add the name of your preselection as a `SEL='PreselXXX>0'` requirement.
  - You can add as many selections as you like combined with $\&\&$ or $||$.
  - It's basically a root `TCut`.

How to write an ETC

```
from Configurables import EventTuple, TupleToolSelResults
tag = EventTuple("TagCreator")
tag.EvtColsProduce = True
tag.ToolList = [ "TupleToolEventInfo", "TupleToolRecoStats",
                 "TupleToolSelResults"  ]
tag.addTool(TupleToolSelResults )
tag.TupleToolSelResults.Selections  = [ "PreselB2DiMuon",
                                        "PreselBd2KstarMuMu" ]

DaVinci().MoniSequence = [ tag ]
DaVinci().ETCFile = "DVPresel_ETC2.root"
```

`EventTuple` now allows to save the Tuple as an ETC

`TupleToolSelResults` stores the `filterPassed` result of any algorithm.

`ETCFile`: Save your tuple. Done.

How to read an ETC

```
DaVinci().Input   = [
    "COLLECTION='TagCreator/EventTuple' DATAFILE='DVPresel_ETC2.root' TYP='PO
from Configurables import TagCollectionSvc
ApplicationMgr().ExtSvc  += [ TagCollectionSvc("EvtTupleSvc") ]
FileCatalog().Catalogs = [ "xmlcatalog_file:options/2008-InclJpsiDst.xml" ]
```

- Declare your ETC as input data
- Declare `"EvtTupleSvc"` (will eventually move to `DaVinci()`).
- Declare the file catalogue
  - Tells **Gaudi** where to find the data to which the ETC points
  - Generated with

```
SetupProject Dirac
lhcb-proxy-init
dirac-lhcb-generate-catalog \
  LFN:/lhcb/MC/2008/DST/00003401/0000/00003401_0000000{1,2}_5.dst
```

- Look at `options/DaVinci-*ETC.py` in **DaVinci**.

How to write a DST

```
from Configurables import DaVinci, DaVinciWriteDst
DaVinciWriteDst().DstFiles[ "Jpsi_1.dst" ] = seq1 # some selection sequence
DaVinciWriteDst().DstFiles[ "Jpsi_2.dst" ] = seq2 # some other selection
DaVinciWriteDst().Items += [ "/Event/Phys/Jpsi_1#2",
                             "/Event/Phys/Jpsi_2#2" ]
seq =  DaVinciWriteDst().dstSequence()
DaVinci().UserAlgorithms = [ seq ]
```

`DaVinciWriteDst().DstFiles` is a dictionary of (file : sequencer).

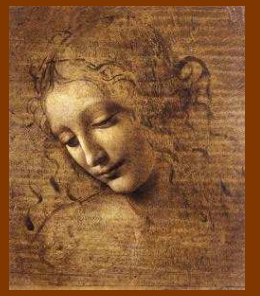
`DaVinciWriteDst().dstSequence()`: Returns one sequence containing all sub-sequences to pass to `DaVinci()` .

`DaVinciWriteDst().Items` allows you to add some locations to the standard DST.

- The same event can go to several DSTs.
- Look at `options/DaVinci-*Dst.py` in **DaVinci**.

What you get

DaVinciMainSequence	8.939	9.452	0.549	726.0
DstWriters	8.939	9.450	0.547	726.0
Seq_3000_3500	1.180	1.211	0.009	103.6
Prescale_3000_3500	0.010	0.013	0.007	0.1
Jpsi_3000_3500	0.188	0.052	0.039	0.1
Print_3000_3500	9.798	9.898	0.093	98.0
Seq_3050_3150	7.689	8.195	0.085	725.7
Prescale_3050_3150	0.000	0.006	0.004	0.0
Jpsi_3050_3150	0.050	0.051	0.036	0.4
Print_3050_3150	0.107	0.144	0.051	0.7
Seq_2600_3200	0.050	0.039	0.007	0.7
Prescale_2600_3200	0.020	0.007	0.005	0.0
Jpsi_2600_3200	0.000	0.041	0.036	0.1
Print_2600_3200	0.000	0.128	0.050	0.5
DSTJpsi_3000_3500	39.054	44.898	0.002	3548.8
DSTJpsi_3050_3150	38.144	52.462	0.001	295.8
DSTJpsi_2600_3200	3.869	3.906	0.001	109.8



`DaVinci()` for experienced

ApplicationMgr().TopAlg

Many problems in options occur because the order in which algorithms are executed is not the one expected.

- `DaVinci()` **resets** `ApplicationMgr().TopAlg`
  - ✗ Whatever `TopAlg` you define in your options is ignored
  - ➜ Use `UserAlgorithms` to run it
  - ✔ This ensures a reasonable ordering of execution
- The order in which configurables are initialised in not predictable (by anyone but Marco)
  - ➜ Avoid using `ApplicationMgr().TopAlg` in sub-configurables
    - They should either populate sequences passed to them
    - Or return already made sequences
    - In both cases it's `DaVinci` which populates the `TopAlg`

How do I run my selection?

If you have a some options that work with a previous version of **DaVinci** (even `v19`) it should take 2 minutes to get it to work with `v22r0`.

- There's no good reason to stick to old versions of **DaVinci**
- There will be no support
- You get an obsolete version of the trigger
- **DaVinci** `v22r0` can do everything previous versions were able to do
- . . . and more
  - You need this version to read the latest **Brunel** data

See `https://twiki.cern.ch/twiki/bin/view/LHCb/DaVinciConfigurable`

A. The local approach

You can put everything in a single option file:

```
from Gaudi.Configuration import *
##########################################
# Define your sequencer
#
from Configurables import GaudiSequencer
mySeq = GaudiSequencer("MySeq")
#
# Add algorithms to your sequencer here:
# mySeq.Members += ....
##########################################
# Configure the job
#
from Configurables import DaVinci
DaVinci().EvtMax = 100                      # Number of events
DaVinci().DataType = "DC06"                 # Default is "DC06"
DaVinci().Simulation   = True               # It is MC
DaVinci().UserAlgorithms = [ mySeq ]        # Declare your sequencer
```

B. Catch the sequencer

You can import an option file, catch the sequencer that is defined in the file and declare it:

```
from Gaudi.Configuration import *
from Configurables import GaudiSequencer
##########################################
# Import two predefined sequences
importOptions("$B2DILEPTONROOT/options/DoDC06SelBu2MuMuK.py")
importOptions("$CCBARROOT/options/DoDC06SelBs2JpsieePhi.py")
##########################################
# Catch the sequencers
mmK = GaudiSequencer("SeqPreselBu2LLK")
eePhi = GaudiSequencer("DC06SelBs2JpsieePhiFilterSequence")
##########################################
# Configure the job
from Configurables import DaVinci
DaVinci().EvtMax = 100                      # Number of events
DaVinci().DataType = "DC06"                 # Default is "DC06"
DaVinci().Simulation   = True               # It is MC
DaVinci().UserAlgorithms = [ mmK, eePhi ]  # Declare your sequencers
```

This is the simplest way to run two selections.

C. Use `MainOptions`

You can run your selection as a “main” option file.

```
from Gaudi.Configuration import *
##########################################
# Configure the job
#
from Configurables import DaVinci
DaVinci().EvtMax = 100           # Number of events
DaVinci().DataType = "DC06"      # Default is "DC06"
DaVinci().Simulation   = True    # It is MC
DaVinci().MainOptions = "$DAVINCIROOT/options/MySequence.py"
```

This also works with text options (`.opts`)

PhysDesktop changes

There are two main changes compared to what has been advertised in the past

- `addTool`: write

```
MyAlg.addTool(PhysDesktop)
# DANGEROUS : MyAlg.addTool(PhysDesktop())
```

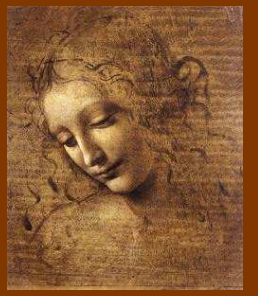
- `InputLocations`: write

```
MyAlg.PhysDesktop.InputLocations = [ 'MyOtherAlg' ]
# Deprecated: MyAlg.PhysDesktop.InputLocations = [ 'Phys/MyOtherAlg' ]
```

similarly:

```
MyHltAlg.Context = 'Hlt'
MyAlg.PhysDesktop.InputLocations = [ 'Hlt2Pions' ]
# Deprecated : MyAlg.PhysDesktop.InputLocations = [ 'Hlt/Hlt2Pions' ]
# DOES NOT WORK : MyAlg.PhysDesktop.InputLocations = [ 'Phys/Hlt2Pions' ]
```

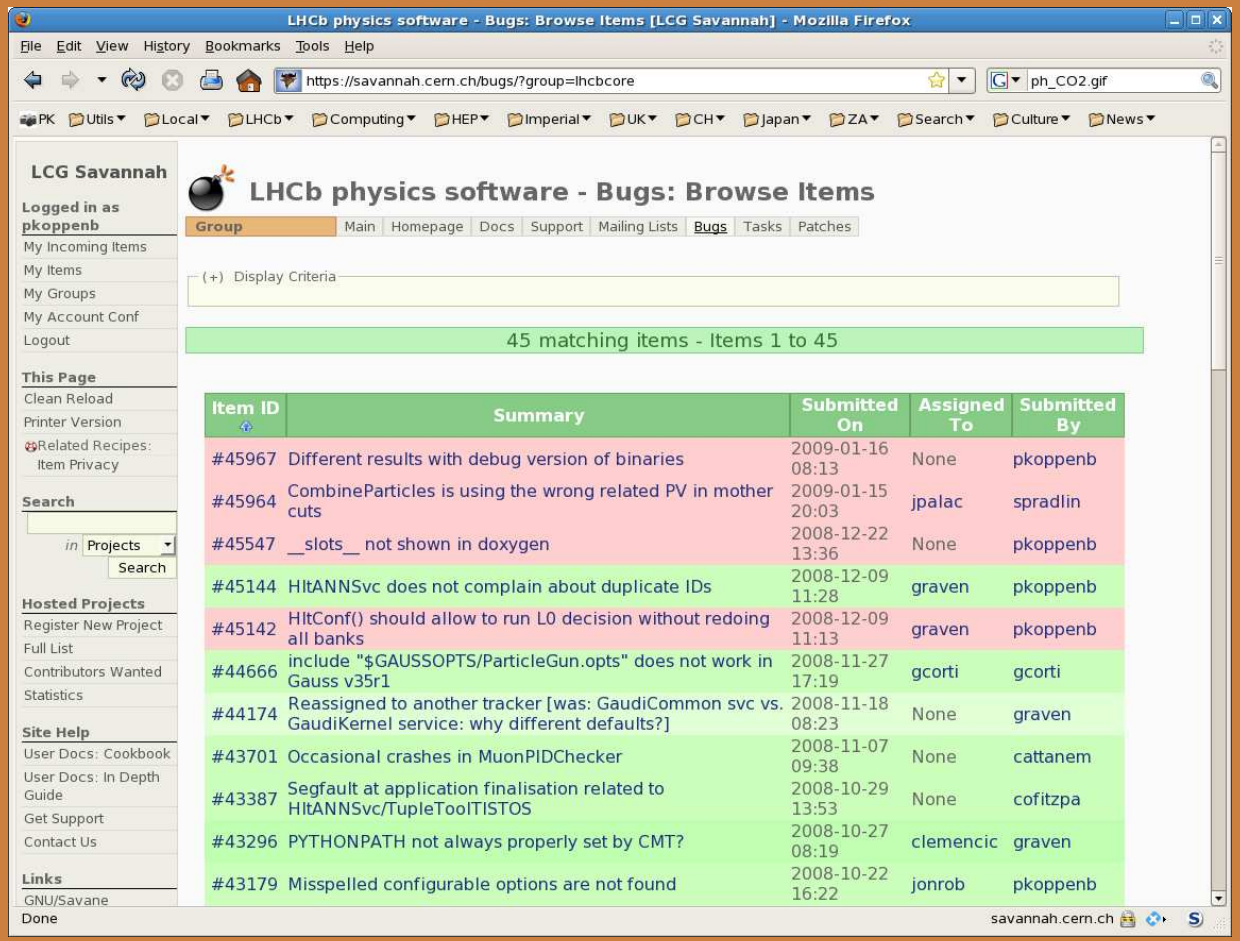
To do list



- Handling of DDDB and CondDB tags
  - Now “default”
  - ➜ Could this come from the bookkeeping?
  - Same for `DaVinci().DataType`
  - Is there an algorithm that stops if the data is not compatible with the given type? That would avoid potential errors.
- MicroDst configurable ? (Juan)
- Rewrite of stripping (Anton)

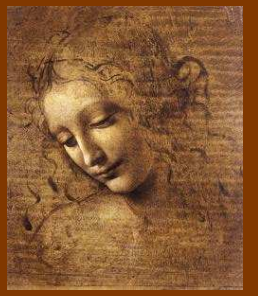
Savannah

- Please check and report bugs in Savannah
- Get a Savannah account
- Become member of the group



https://savannah.cern.ch/bugs/?group=lhcbcore

Conclusion



- `DaVinciCommon.opts` is gone
  - ➜ You have to do something to get your options to work
    - It's not difficult
- DST/ETC writing restored
- Only the new version can read new **Brunel** data
- You need it to get developments of the HLT
- ➜ Please use it and report problems
- Suggestions are welcome