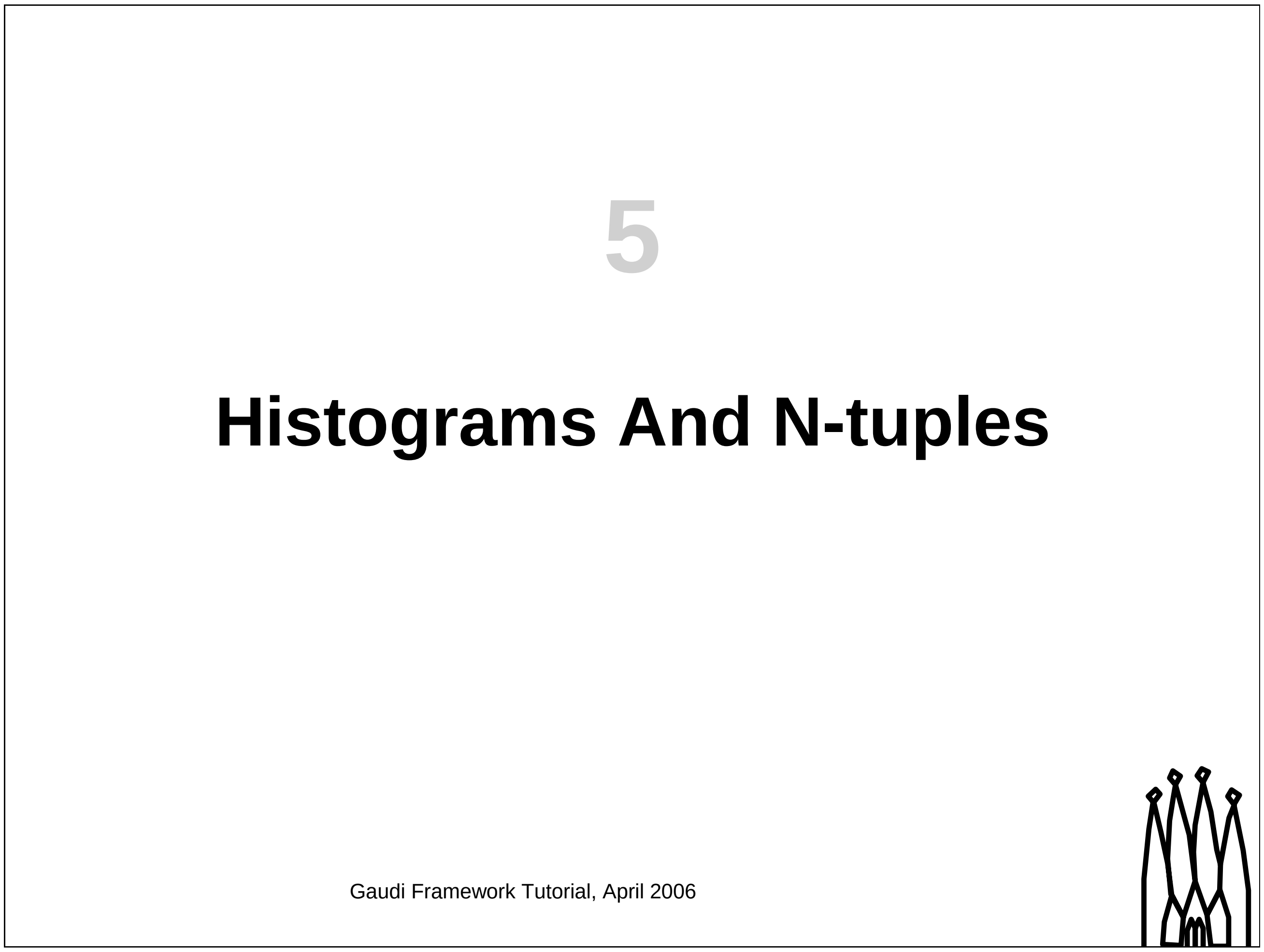


5

Histograms And N-tuples

Gaudi Framework Tutorial, April 2006

Schedule:	Timing	Topic
	15 minutes	Lecture
	20 minutes	Practice
	35 minutes	Total

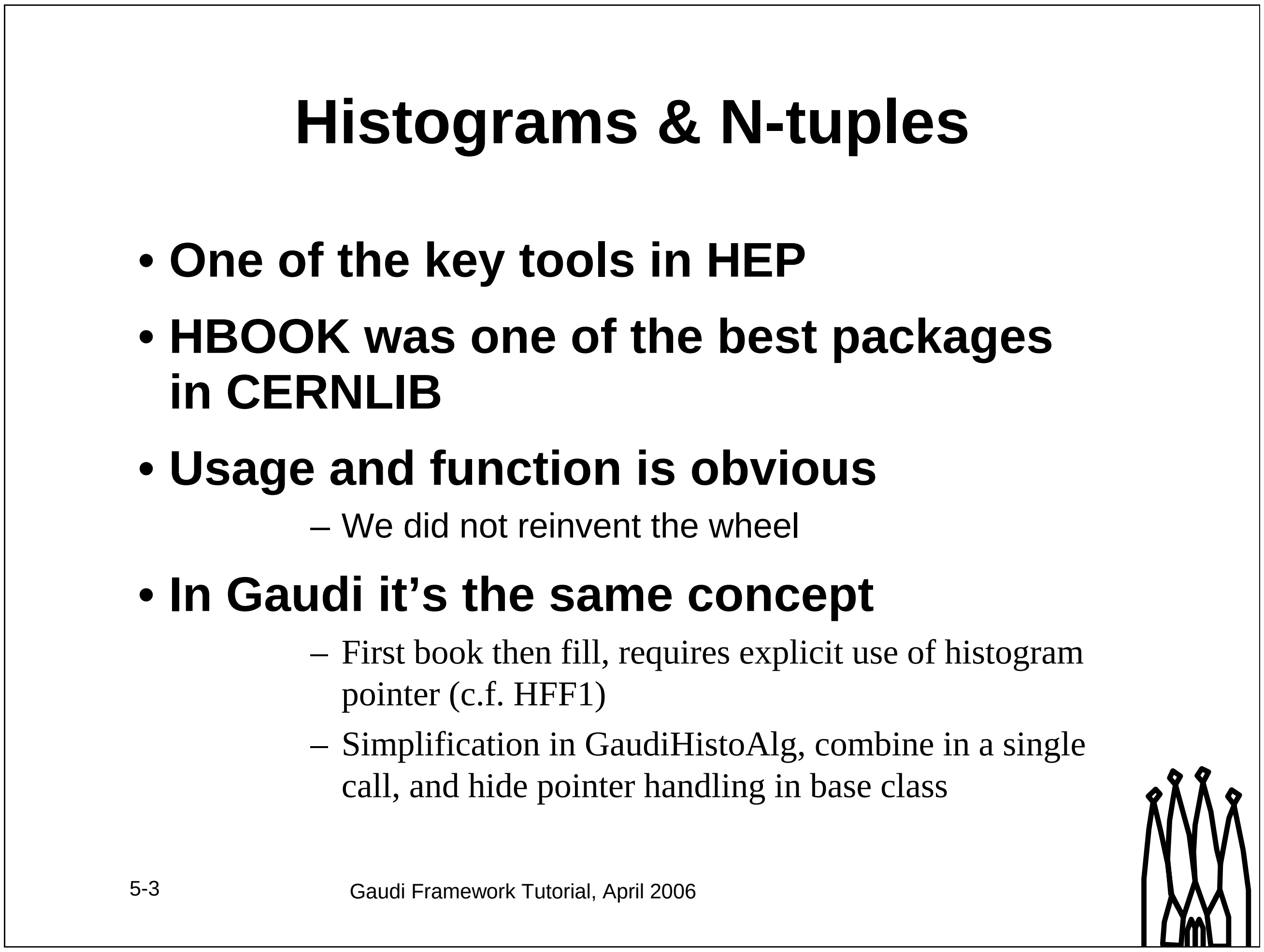
Objectives

After completing this lesson, you should be able to:

- **Book and fill histograms.**
- **Book and fill N-tuples.**

Lesson Aim

The aim of this topic is to introduce histograms and N-tuples as they were introduced in Gaudi. It will be shown how to book histograms and N-tuples and how to fill them.



Histograms & N-tuples

- One of the key tools in HEP
- HBOOK was one of the best packages in CERNLIB
- Usage and function is obvious
  - We did not reinvent the wheel
- In Gaudi it’s the same concept
  - First book then fill, requires explicit use of histogram pointer (c.f. HFF1)
  - Simplification in GaudiHistoAlg, combine in a single call, and hide pointer handling in base class

5-3 Gaudi Framework Tutorial, April 2006

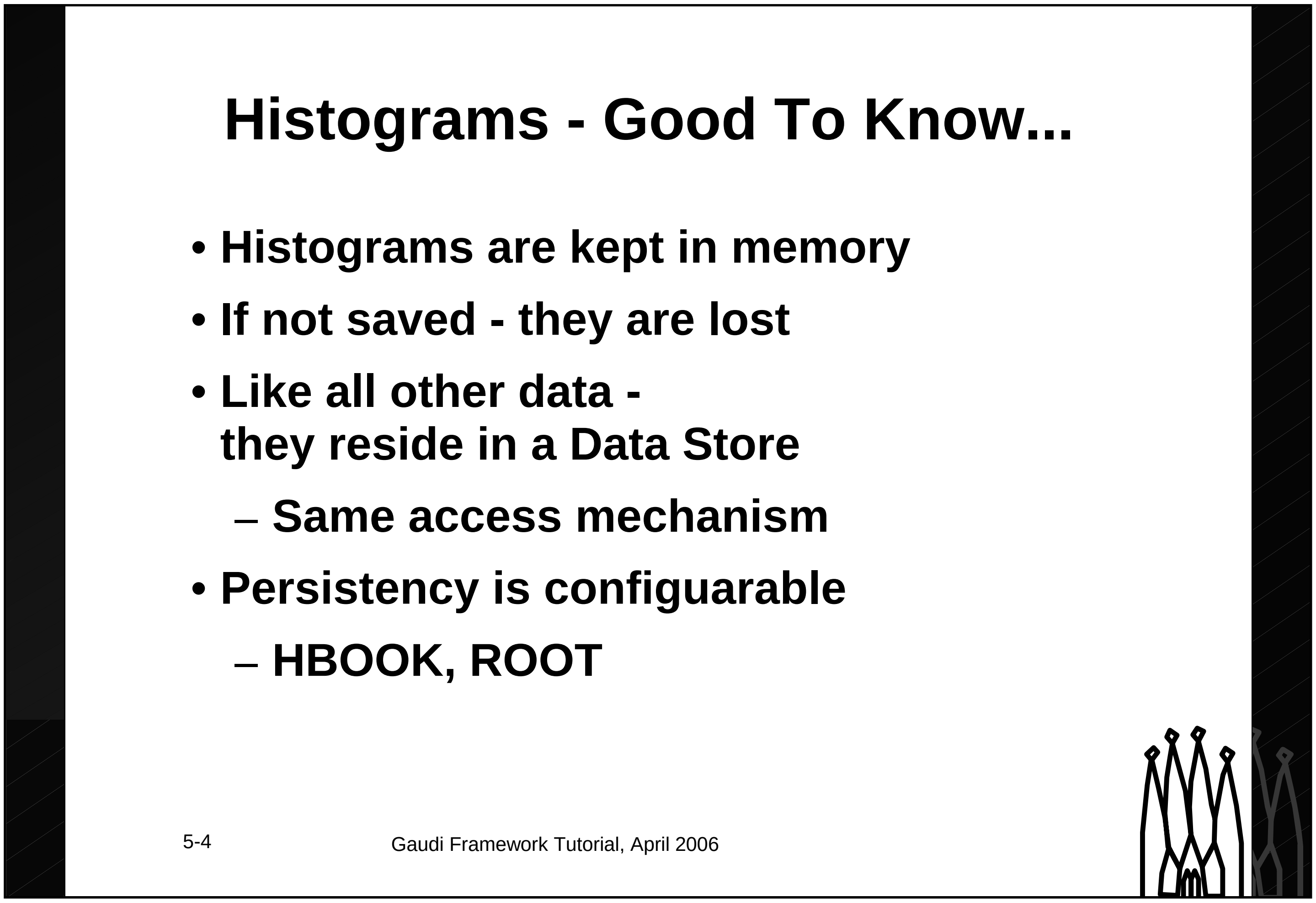
Histograms & N-tuples

Histograms and N-tuples are one of the key concepts for data analysis in high energy physics. In fact, HBOOK was one of the most successful packages in CERNLIB, both for batch processing and interactive use within PAW.
The usage of HBOOK was obvious. Within Gaudi the wheel was not re-invented, but is rather based on the known concepts:

•The histogram or the N-tuple must be booked

•Then it can be filled.

We have simplified this by merging the two functions into a single call in the GaudiHistoAlg base class



Histograms - Good To Know...

- Histograms are kept in memory
- If not saved - they are lost
- Like all other data -
  they reside in a Data Store
  – Same access mechanism
- Persistency is configuarable
  – HBOOK, ROOT

5-4 Gaudi Framework Tutorial, April 2006

Histograms - Good To Know

Like in the old CERNLIB approach histograms are kept in memory. This allows fast access for filling and other operations without unnecessary I/O.

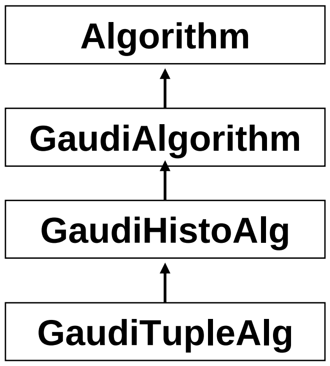
However, this means that the histograms must be saved at the end of the job. Otherwise the histograms are lost as soon as the process terminates.

Histograms within Gaudi reside on a data store - similar to event data. This allows to use the same access mechanism.

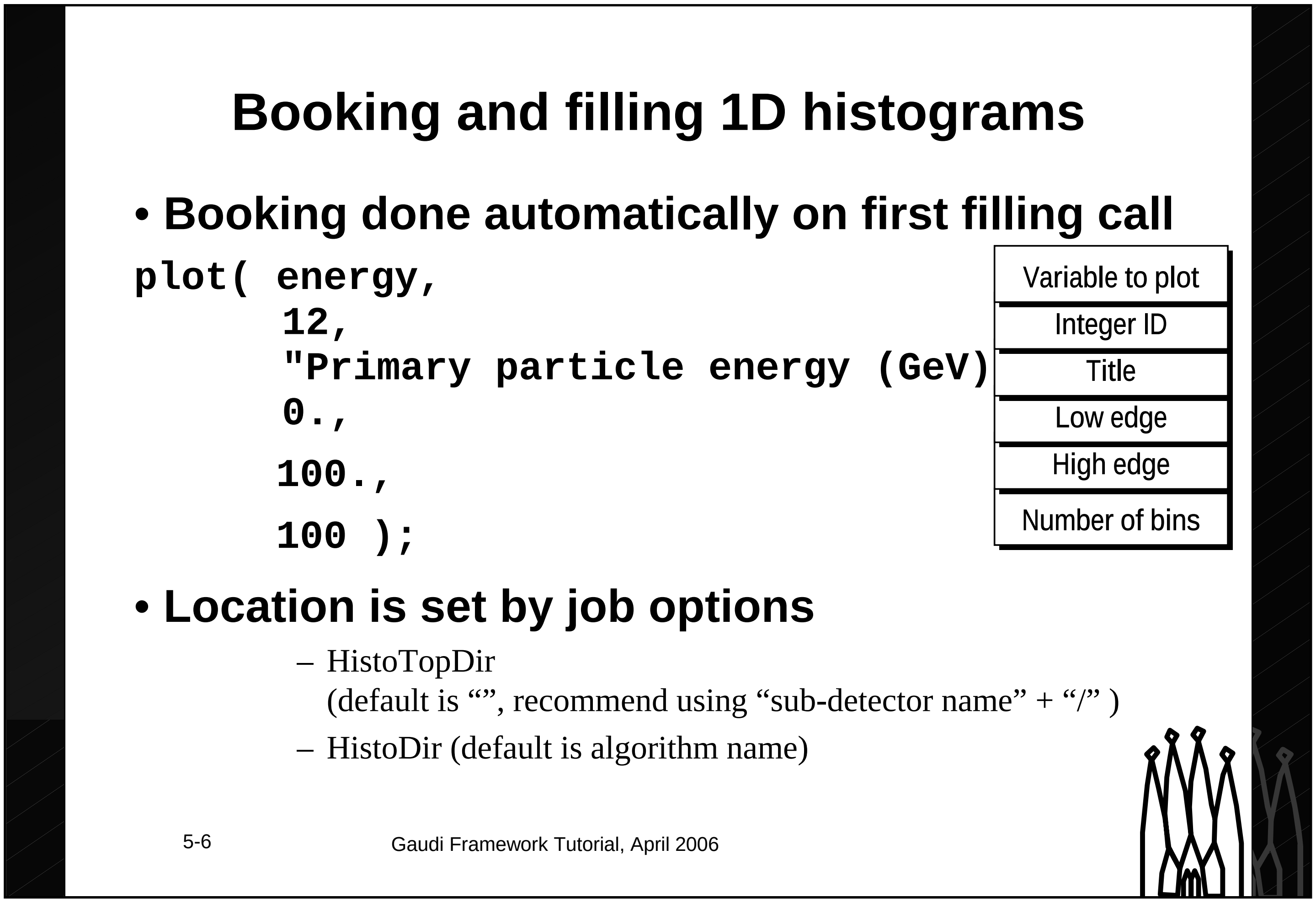
The persistency mechanism for histograms in Gaudi is configurable. Persistent back-ends exist using HBOOK or ROOT. However, the HBOOK one is no longer supported. you are strongly encouraged to use ROOT.

GaudiHistoAlg and GaudiTupleAlg

- Specialisations of GaudiAlgorithm



- Simplify handling of histograms and N-tuples



Booking and filling 1D histograms

• **Booking done automatically on first filling call**

```
plot( energy,
      12,
      "Primary particle energy (GeV)
      0.,
      100.,
      100 );
```

Variable to plot

Integer ID

Title

Low edge

High edge

Number of bins

• **Location is set by job options**

– HistoTopDir (default is “”, recommend using “sub-detector name” + “/”)

– HistoDir (default is algorithm name)

5-6 Gaudi Framework Tutorial, April 2006

Booking and filling 1-d Histograms

The booking of histograms in Gaudi is delegated to the histogram service, which is in charge of managing the corresponding transient data store. The service returns a pointer to the booked histogram, that then has to be cached by the calling algorithm and used when filling (hbook style book and fill methods).

GaudiHistoAlg hides these technicalities. The plot() method combines the booking and filling functionality; the histogram is booked on the first request to fill it.

The first argument is the variable to plot. The remaining arguments are needed for booking (note the different order of the arguments compared to HBOOK):

•Integer ID is mandatory only if you want to save the histograms in HBOOK. If missing, the histogram can only be saved in Root, and is identified by its title.

•the histogram title

•the lower edge of the histogram

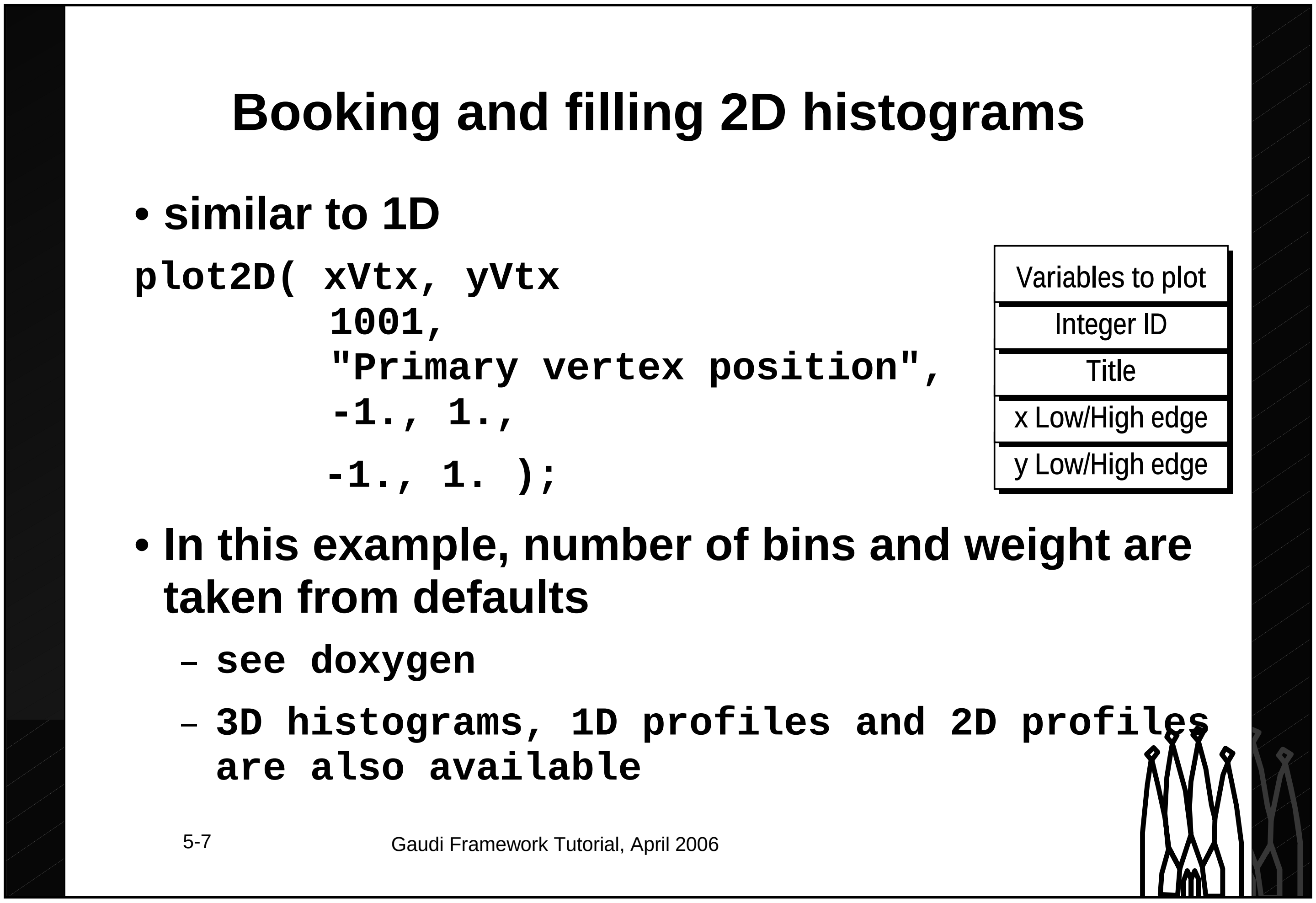
•the upper edge of the histogram

•The number of bins (optional, defaults to 100)

•An optional weight

The histograms are located in directories, one per algorithm. The full path is

“/stat/” + HistoTopDir + HistoDir



Booking and filling 2D histograms

- similar to 1D

```
plot2D( xVtx, yVtx
        1001,
        "Primary vertex position",
        -1., 1.,
        -1., 1. );
```

Variables to plot
Integer ID
Title
x Low/High edge
y Low/High edge

- In this example, number of bins and weight are taken from defaults
  - see doxygen
  - 3D histograms, 1D profiles and 2D profiles are also available

5-7 Gaudi Framework Tutorial, April 2006

Generic histogram access

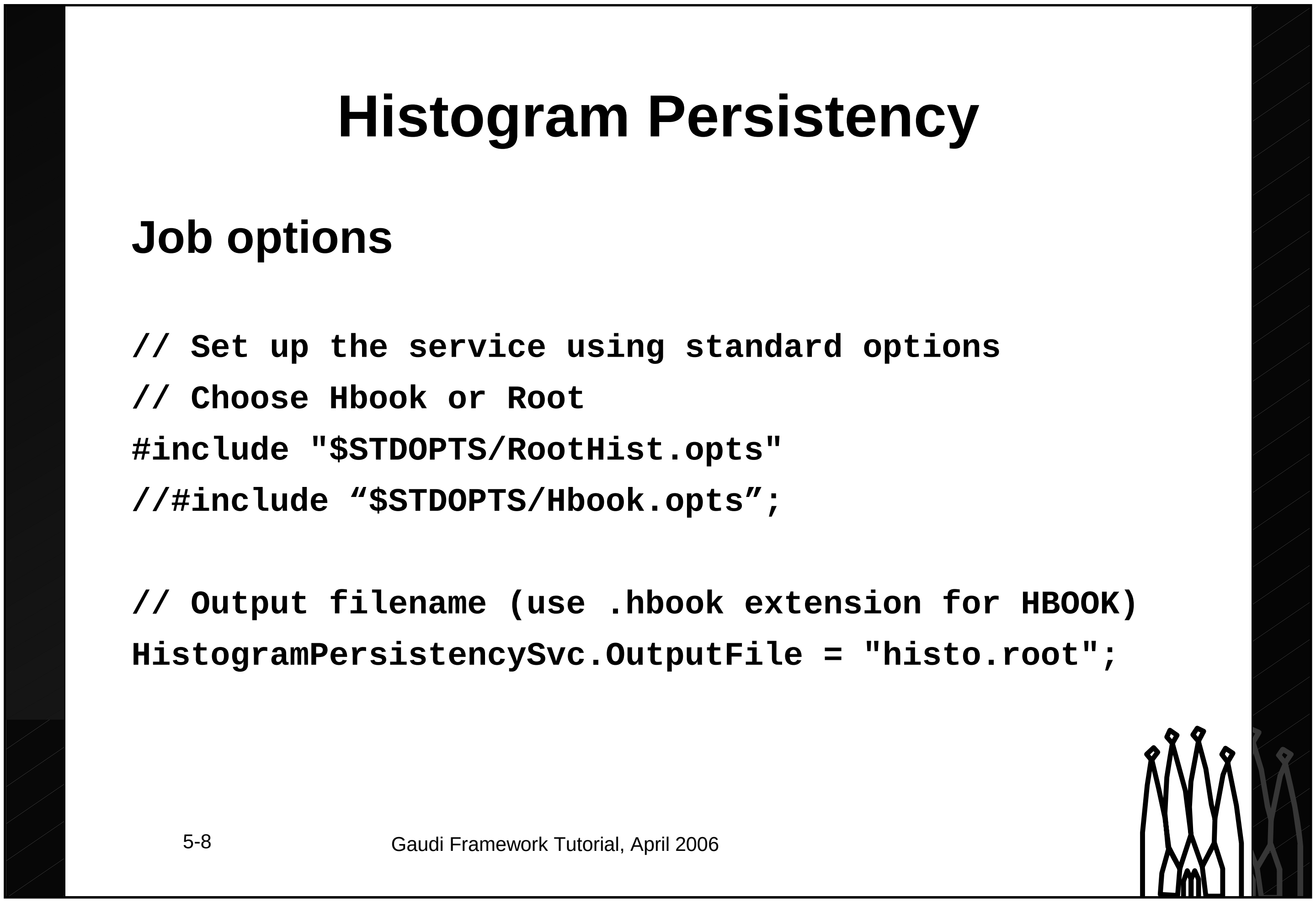
Histograms are data store objects like any other. So they can be accessed like any other data store object:

```
SmartDataPtr<IHistogram1D>
  h1d(histoSvc(), "simple/3");
  if ( h1d ) {
    h1d->fill( value, weight);
  }
```

Note on efficiency

The plot methods shown in the last two slides keep a hash table internally to lookup the histogram pointer. For fast access in time critical applications, it is recommended to cache the pointer returned by the first call to the plot method, and to use directly the fill method shown above in subsequent calls.

Note also that locating a histogram via a SmartDataPtr introduces an additional overhead compared to the usage of the pointer directly. However, this usage sometimes is necessary e.g. if the histogram was booked in one algorithm, but has to be manipulated in another algorithm. However, this should only be used for short term tests, because it also introduces run-time coupling.



Histogram Persistency

Job options

```
// Set up the service using standard options
// Choose Hbook or Root
#include "$STDOPTS/RootHist.opts"
//#include “$STDOPTS/Hbook.opts”;

// Output filename (use .hbook extension for HBOOK)
HistogramPersistencySvc.OutputFile = "histo.root";
```

5-8 Gaudi Framework Tutorial, April 2006

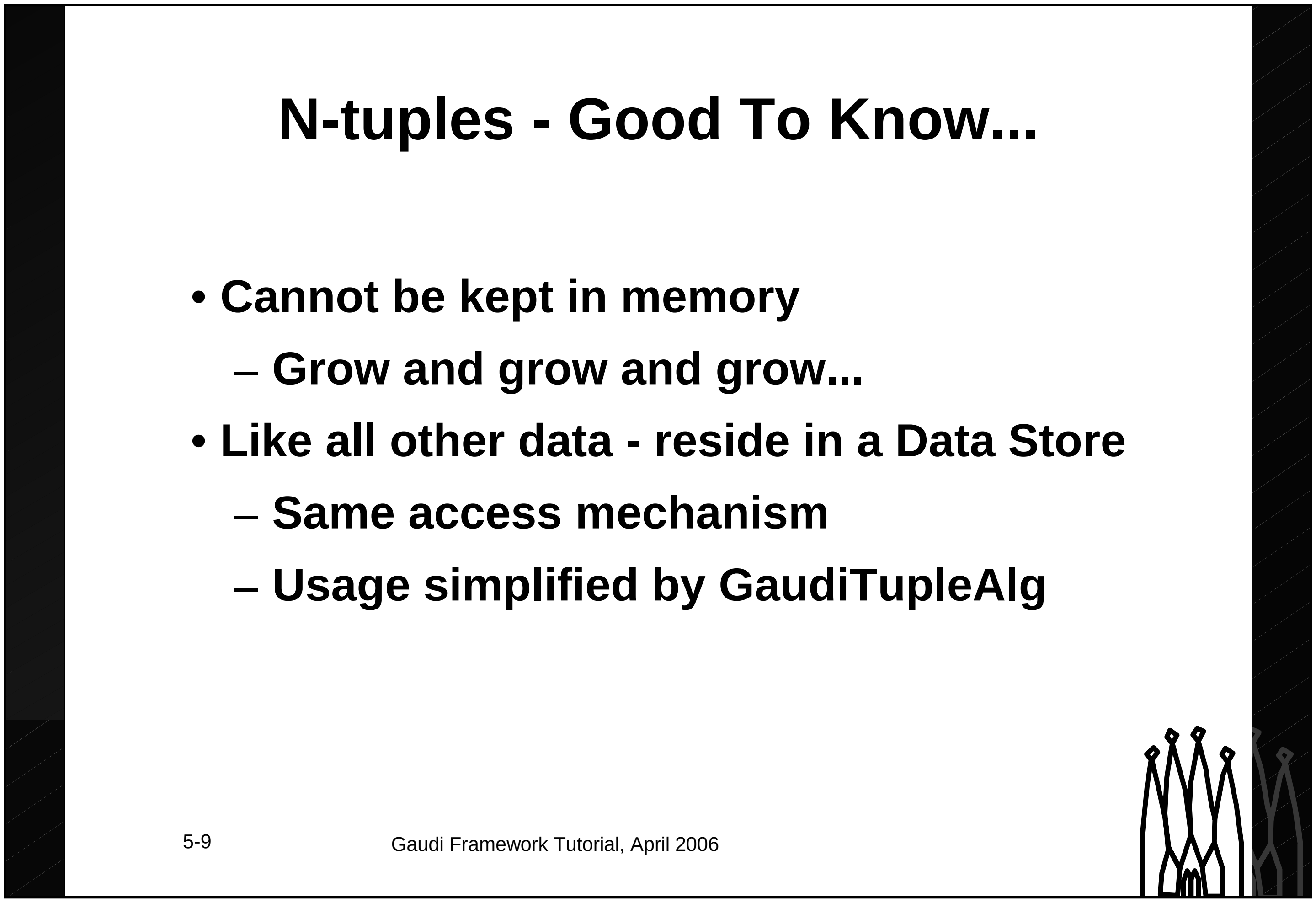
Histogram Persistency

The persistent back-end for the histograms is defined in the job options file. To simplify things, standard options files are provided to set up the service, choose either RootHist.opts or Hbook.opts.

If you do not wish to save the histograms, do not include either file, and specify instead

ApplicationMgr.HistogramPersistency = “NONE";

If histogram persistency is desired the name of the output file must be specified.



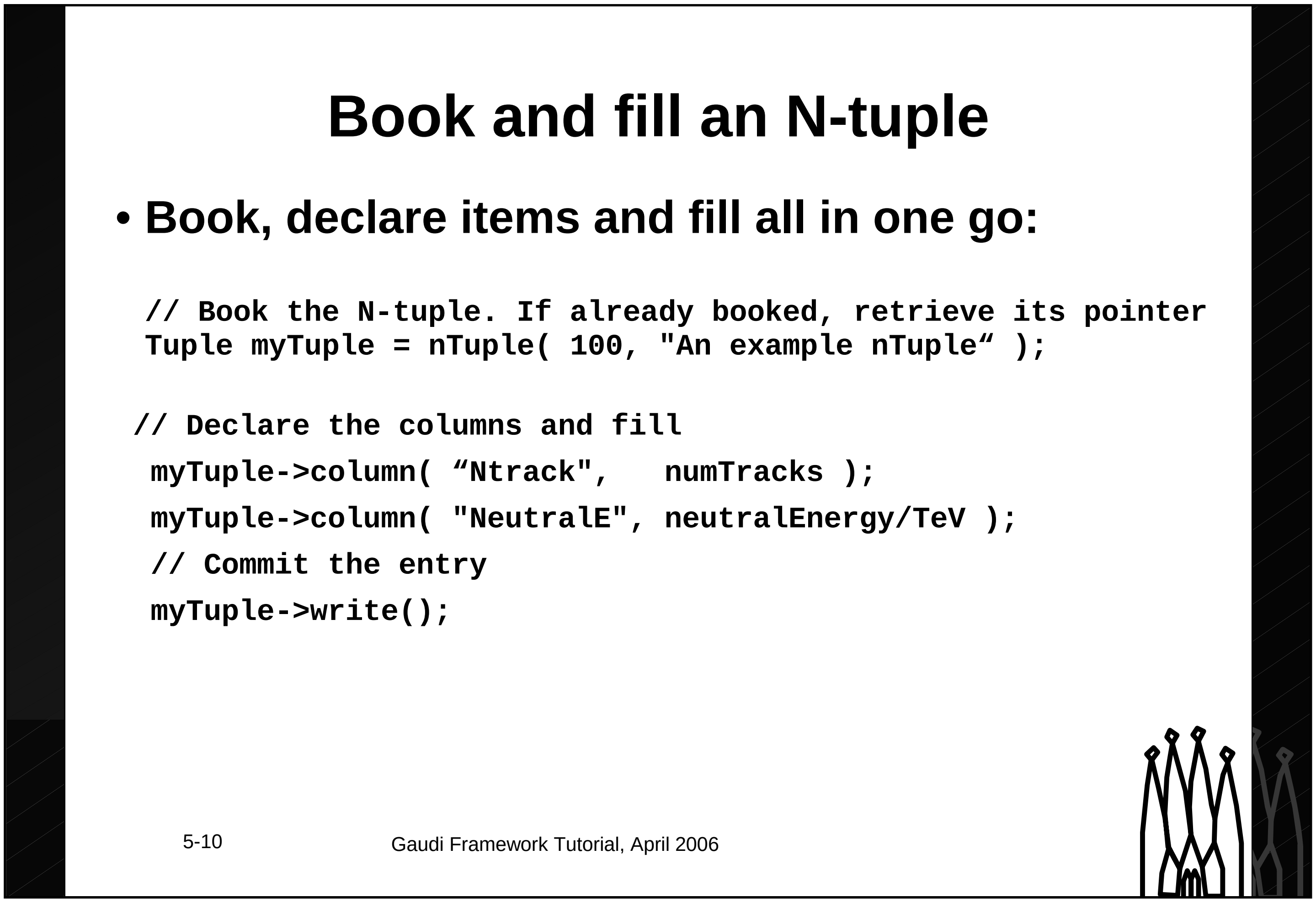
N-tuples - Good To Know...

- Cannot be kept in memory
  - Grow and grow and grow...
- Like all other data - reside in a Data Store
  - Same access mechanism
  - Usage simplified by GaudiTupleAlg

5-9 Gaudi Framework Tutorial, April 2006

N-tuples - Good to know

N-tuples - unlike histograms - cannot be kept in memory. Because there are constantly rows added the size of the N-tuple is not constant. They grow and grow. N-tuples must be connected to a persistent back-end already when filled.
Like all other data, N-tuples are also data store objects.



Book and fill an N-tuple

- **Book, declare items and fill all in one go:**

```
 // Book the N-tuple. If already booked, retrieve its pointer
 Tuple myTuple = nTuple( 100, "An example nTuple“ );

// Declare the columns and fill
  myTuple->column( “Ntrack",   numTracks );
  myTuple->column( "NeutralE", neutralEnergy/TeV );
  // Commit the entry
  myTuple->write();
```

5-10 Gaudi Framework Tutorial, April 2006

Book the N-tuple

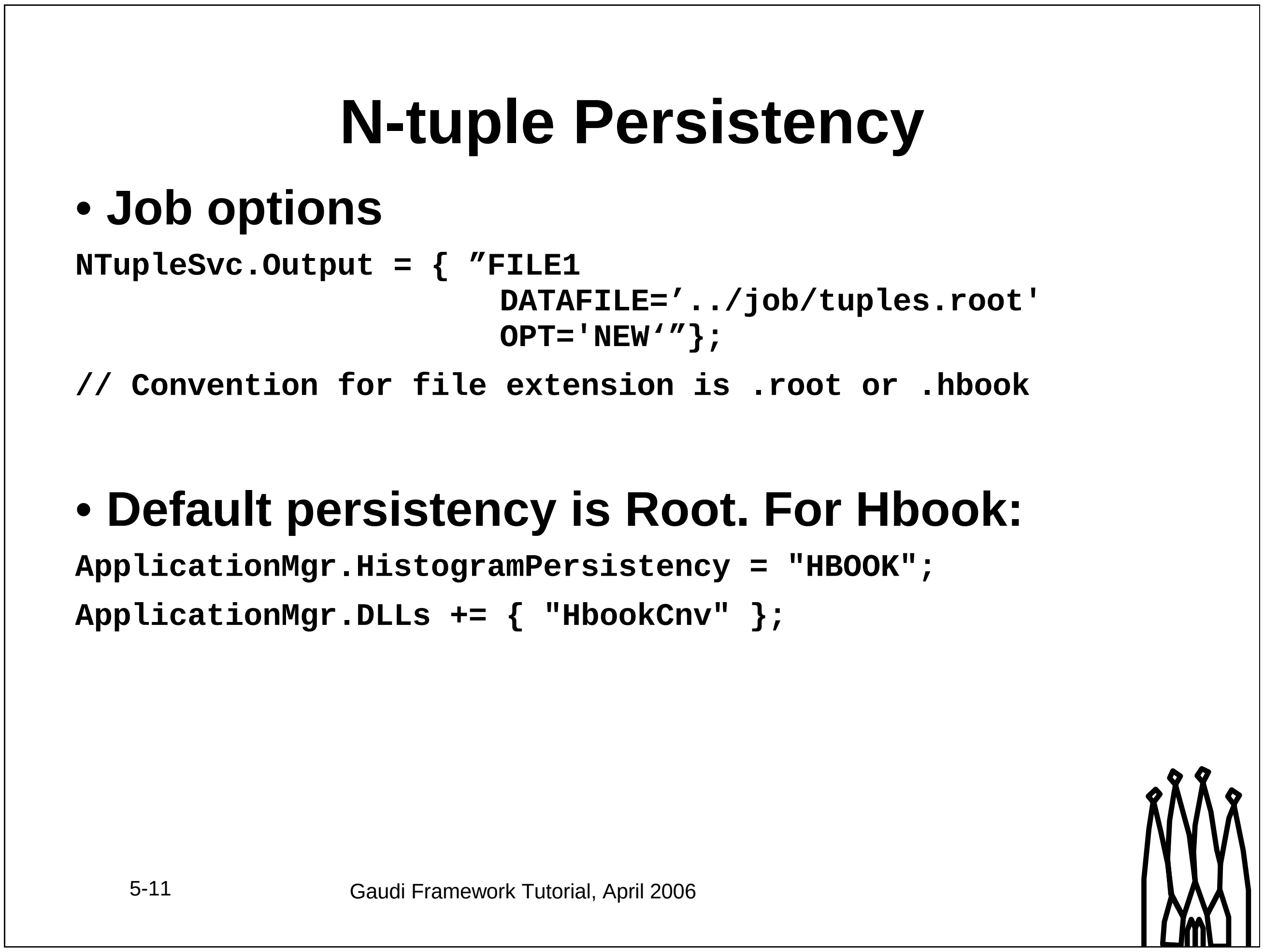
To book an N-tuple, specify the N-tuple identifier, the title and, optionally, the type (the default is CLID_ColumnWiseTuple, CLID_RowWiseTuple is also available) . The identifier itself must obey the same constraints as histograms: if the N-tuple is going to be saved using HBOOK, it must be a number. The N-tuple directory can be controlled by job options, “NTupleTopDir” and “NTupleDir” with the same defaults as for histograms. The “NTupleLUN” can also be specified, to write N-tuples from different algorithms to different files. Default is “FILE1”

Define and fill N-tuple Columns

Once you have the Tuple pointer, you can declare its columns. Declaration and filling of columns is done in a single call, the first argument is the Column name, the second the value to enter in the N-tuple.

Write the N-tuple

Once all items are assigned to the N-tuple the tuple can be written to disk. This is done by calling the write() method



N-tuple Persistency

- **Job options**

```
NTupleSvc.Output = { ”FILE1
                      DATAFILE=’../job/tuples.root'
                      OPT='NEW‘”};
// Convention for file extension is .root or .hbook
```

- **Default persistency is Root. For Hbook:**

```
ApplicationMgr.HistogramPersistency = "HBOOK";
ApplicationMgr.DLLs += { "HbookCnv" };
```

5-11 Gaudi Framework Tutorial, April 2006

N-tuple Persistency

Output-files for N-tuples are specified in the job-options, similar to the specification of event data input. The first tag is the logical name of the file.

Note:

•“FILEn” is the logical name you can refer to the file within Gaudi. You could use any other name as well like “ECAL” or “Frank”.

Histogram and Ntuple IDs

Numerical and Alpha Numerical IDs possible

E.g.

```
plot1D( energy, 12, "Particle Energy", 0,100,100 );
```

“HistoTopDir/HistoDir/12”

```
plot1D( energy, "en1", "Particle Energy", 0,100,100 );
```

“HistoTopDir/HistoDir/en1”

```
plot1D( energy, “subdir/12", "Particle Energy", 0,
  100, 100 );
```

“HistoTopDir/HistoDir/subdir/12”

```
plot1D( energy, “subdir/en1", "Particle Energy", 0,
  100, 100 );
```

“HistoTopDir/HistoDir/subdir/en1”

Hands On

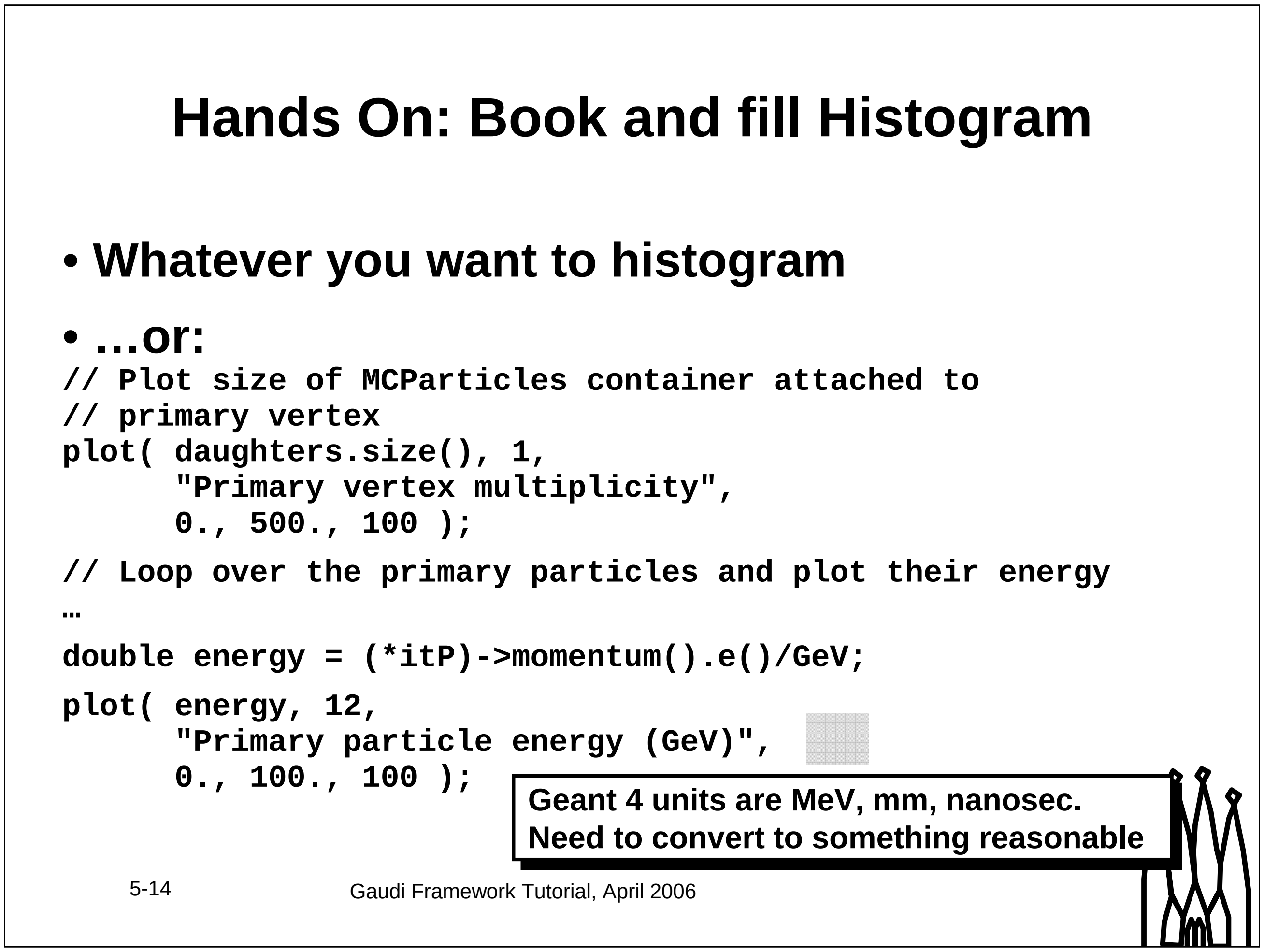
Create a new algorithm

- **VisibleEnergyAlgorithm.h/cpp**
  - **Inheriting from GaudiTupleAlg**
- **Don’t forget to add it to Components_load.cpp**

Include additional header files in .cpp

```
// Event model
#include "Event/MCVertex.h"
#include "Event/MCParticle.h"
#include “Kernel/PhysicalConstants.h“
```

Copy loop over MCParticles from previous example…



Hands On: Book and fill Histogram

- **Whatever you want to histogram**
- **…or:**

```
// Plot size of MCParticles container attached to
// primary vertex
plot( daughters.size(), 1,
      "Primary vertex multiplicity",
      0., 500., 100 );

// Loop over the primary particles and plot their energy
…

double energy = (*itP)->momentum().e()/GeV;

plot( energy, 12,
      "Primary particle energy (GeV)",
      0., 100., 100 );
```

Geant 4 units are MeV, mm, nanosec.
Need to convert to something reasonable

5-14 Gaudi Framework Tutorial, April 2006

Hands On: Book and fill N-tuple

```
Tuple myTuple = nTuple( 100, "An example nTuple",
                        CLID_ColumnWiseTuple );
myTuple->column( "numNeutr", daughters.size()-numCharged );
myTuple->column( "NeutralE", neutralEnergy/TeV );
myTuple->write();
```

Visualize Histograms & N-tuple

…e.g. using ROOT

- **root**
- **root [0] TBrowser B**
- **.. and use menus to open and browse the histogram and Ntuple files**

Solution

In src.histo_ntuple directory of Tutorial/Components package

To try this solution and start next exercise from it:

Uncomment Tutorial 2 options in $MAINROOT/options/jobOptions.opts

```
cd ~/cmtuser/Tutorial/Component/v7r0/src
```

Move your modified files if you want to keep them

```
cp ../src.histo_ntuple/*.* .
cd ../cmt
gmake
$MAINROOT/$CMTCONFIG/Main.exe $MAINROOT/options/jobOptions.opts
```