The CMS Software Framework CMSSW

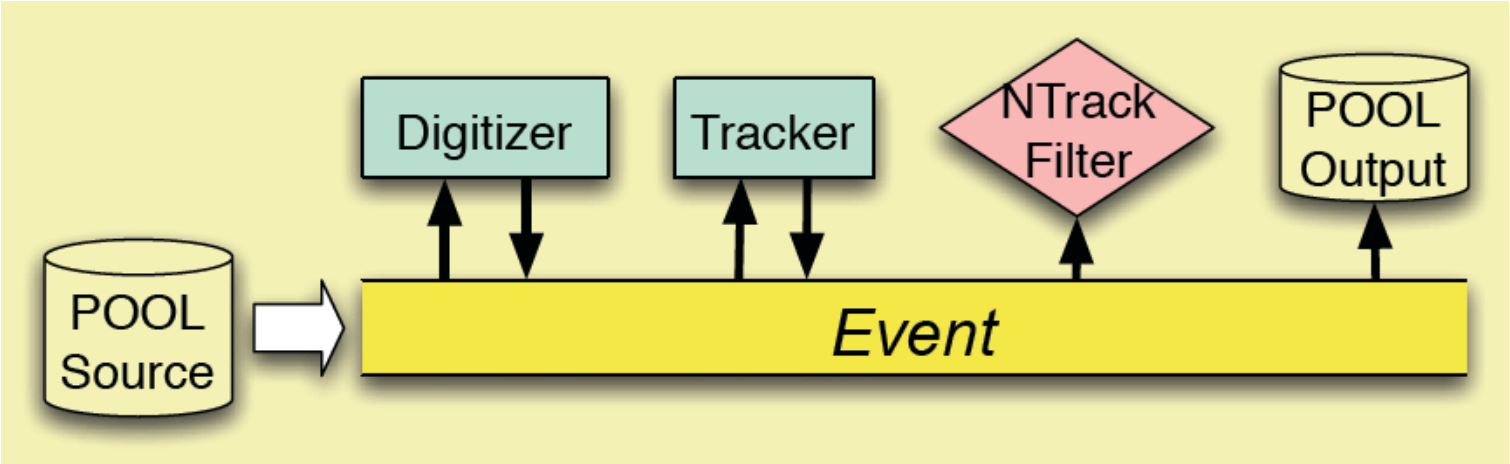
Benedikt Hegner

Outline

- Framework basics
- The Data format
- Writing a framework module
- Running the module
- Places to look for information

The CMSSW framework

The flow of the data



- Source
- EDProducer
- EDFilter
- EDAnalyzer
- OutputModule
- ...

communication between modules only via the event

Steering file for cmsRun

```
process EXAMPLE = {
        source = EmptySource { untracked int32 maxEvents = 2}
        module int = IntProducer { int32 ivalue = 2 }

        module test = IntTestAnalyzer {
                untracked int32 valueMustMatch = 2
                untracked string moduleLabel = "int"
        }
        service = Tracer {}

        path p = {int, test}
}
```

actually used modules (services run automatically!)

```
$ cmsRun example.cfg
```

(with a 2_1_X version it will look differently)

CMSSW file input and output

input

```
source = PoolSource
{
  untracked vstring fileNames = {"file:input.root"}
}
```

PROCESS

output

```
module Out = PoolOutputModule
{
  untracked string fileName = "output.root"
}
```

```
untracked vstring outputCommands =
  {
    "keep *",
    "drop *_*_*_HLT",
    "keep FEDRawDataCollection_*_*_*"
  }
```

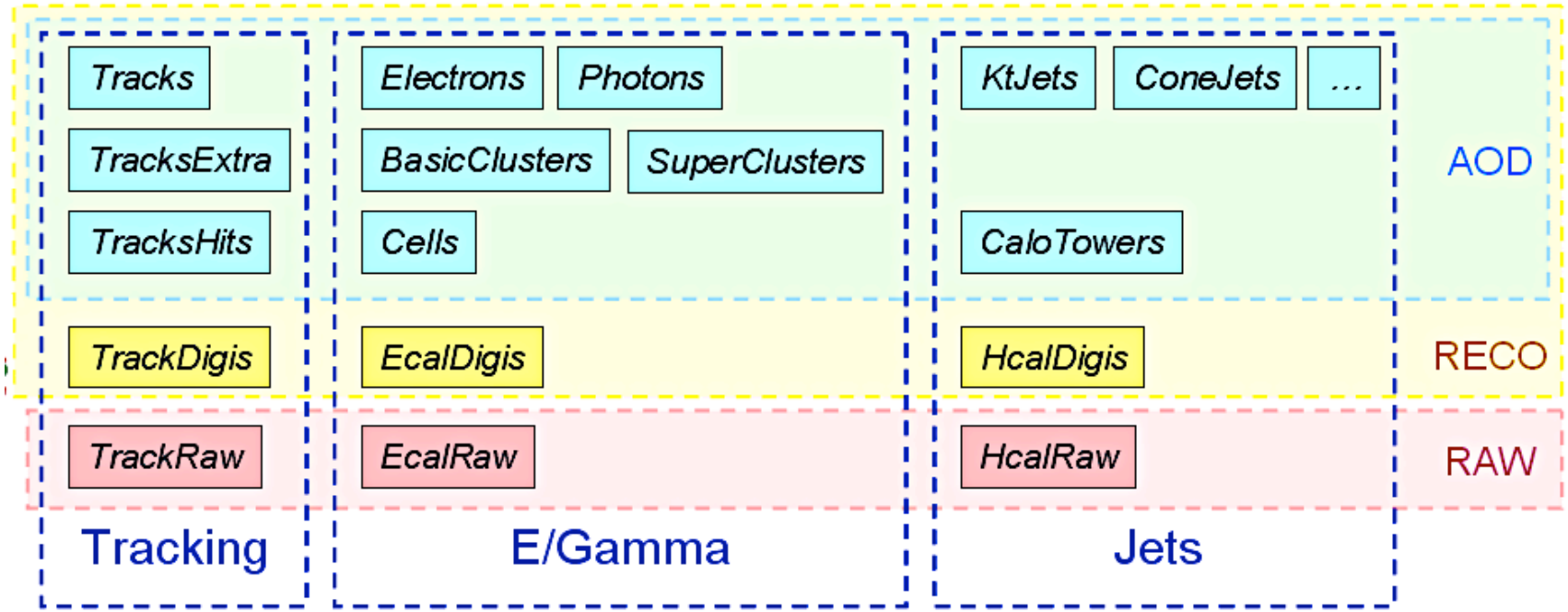
The Data format

What is stored in the event files?

Different file content types

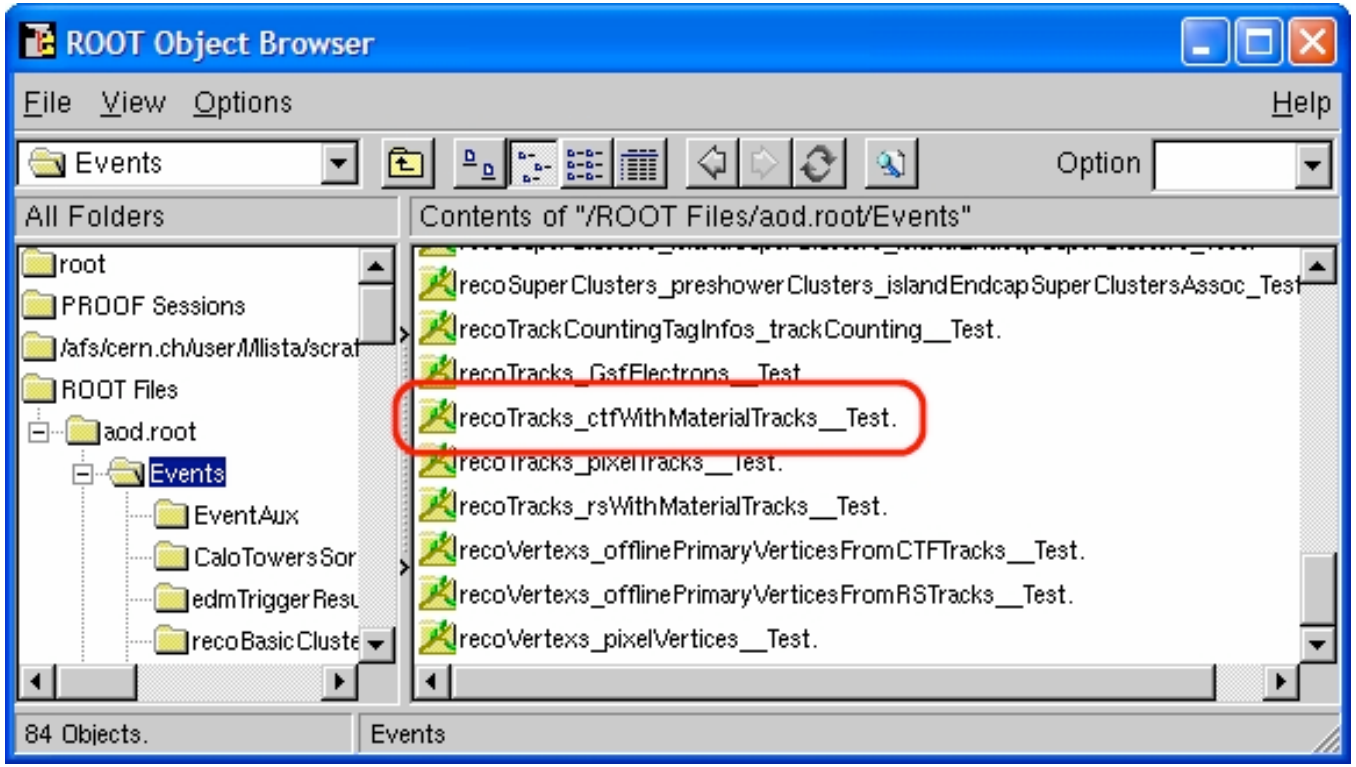
- FEVT: Full EVenT
- RECO: RECOnstruction
- RECOSIM: RECOnstruction + selected simulation information
- AOD: Analysis Object Data (a compact subset of RECO format)
- AODSIM: AOD + generator information

...

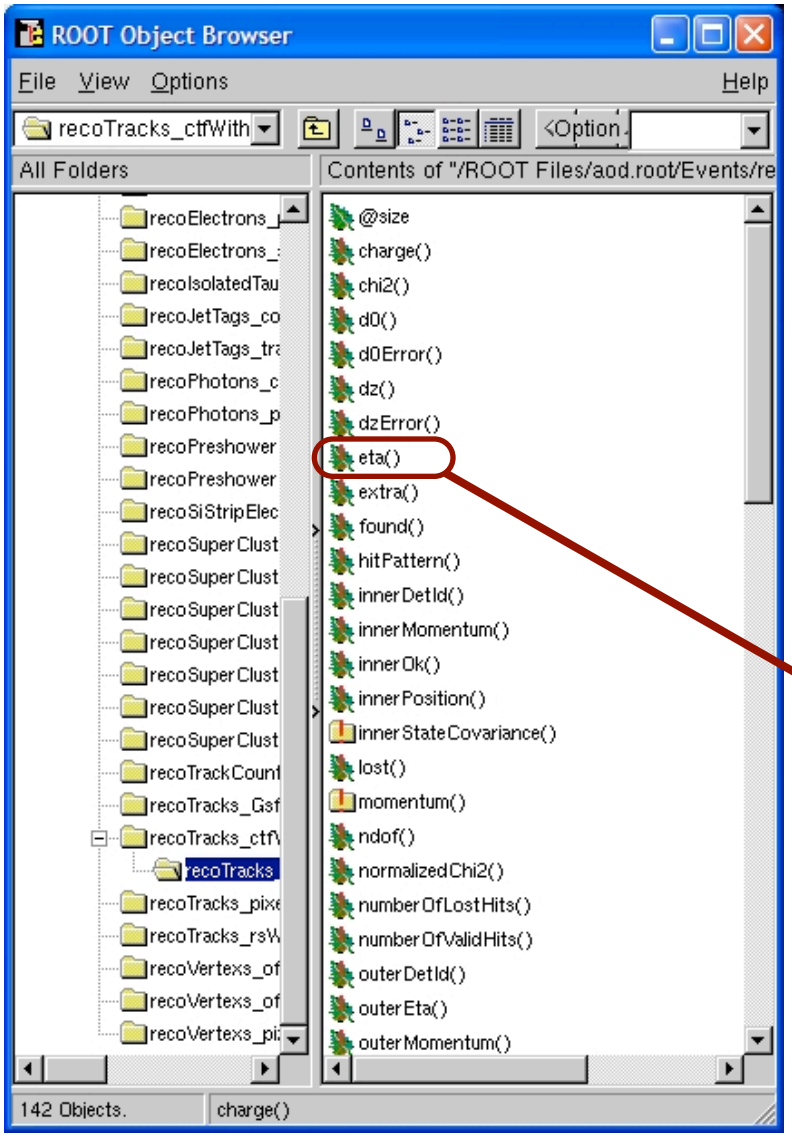


files can be opened with ROOT

```
root.exe
[] TFile f("aod.root")
[] new TBrowser()
```

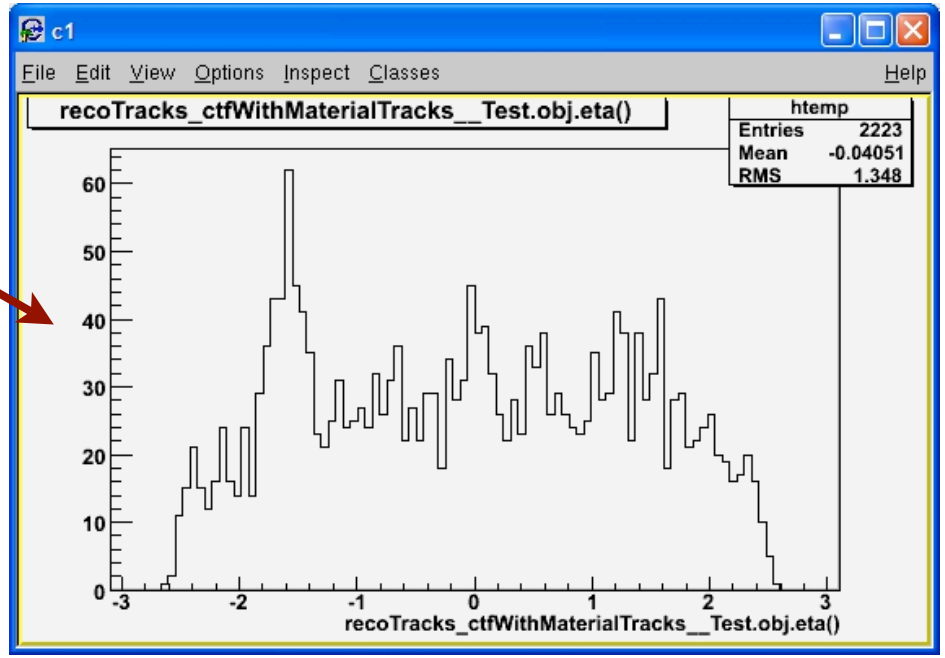


files can be read via FWLite



Automatic library loading

```
[] gSystem->Load("libFWCoreFWLite");
[] AutoLibraryLoader::enable();
[] new TBrowser();
```



Data inside the event are called “Product”

moduleLabel : productInstanceLabel : processName

```
# by module and default product label
 Handle<TrackVector> trackPtr;
 iEvent.getByLabel("tracker", trackPtr );

# by module and product label
 Handle<SimHitVector> simPtr;
 iEvent.getByLabel("detsim", "pixel" ,simPtr );

# by type
 vector<Handle<SimHitVector> > allPtr;
 iEvent.getByType( allPtr );

# by Selector
 ParameterSelector<int> coneSel("coneSize",5);
 Handle<JetVector> jetPtr;
 iEvent.get( coneSel, jetPtr );
```

Output of module EventContentAnalyzer:

```
++Event     0 contains 161 products with friendlyClassName, moduleLabel and productInstanceName:
...
++recoElectrons "siStripElectronToTrackAssociator" "siStripElectrons"
++recoGenJets "Fastjet10GenJets" ""
++recoGenJets "Fastjet10GenJetsNoNu" ""
++recoGenJets "Fastjet6GenJets" ""
++recoGenJets "Fastjet6GenJetsNoNu" ""
++recoGenJets "Kt10GenJets" ""
++recoGenJets "Kt10GenJetsNoNu" ""
++recoGenJets "iterativeCone5GenJets" ""
++recoGenJets "iterativeCone5GenJetsNoNu" ""
++recoGenJets "iterativeCone7GenJets" ""
++recoGenJets "iterativeCone7GenJetsNoNu" ""
++recoGenMETs "genMet" ""
++recoGenMETs "genMetNoNu" ""
...
```

```
Handle<GenJetCollection> genJets;
Event.getByLabel("FastJet6GenJets",genJets );
```

another option:
EdmDumpEventContent <filename>

The history of each single product in the event is stored in the "provenance"

```
...

Module: caloTowers Rec
 PSet id:e03ccfff88a2fd4ed3c2b9bd8261000b
 products: {
  recoCandidatesOwned_caloTowers__Rec.
}
 parameters: {
  @module_label: string tracked  = 'caloTowers'
  @module_type: string tracked  = 'CaloTowerCandidateCreator'
  minimumE: double tracked  = -1
  minimumEt: double tracked  = -1
  src: InputTag tracked  = towerMaker::
}

...
```

edmProvDump <filename>

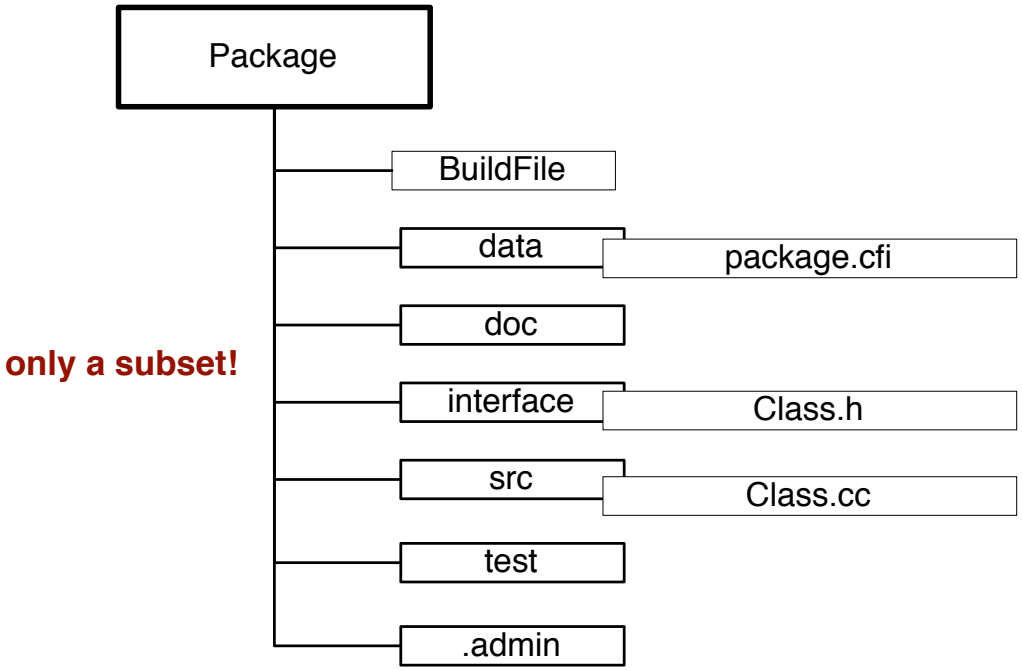
Preparing the environment

```
creating your local area
 $ cmsrel CMSSW_1_6_12
 [...]
 $ cd CMSSW_1_6_12/src

setting runtime variables
 $ cmsenv

accessing the cvs server
 $ addpkg Subproject/Package
```

The release area

SCRAM = Software release and management tool

```
Project/Subproject/Package

CMSSW_1_6_12/src/GeneratorInterface/ToprexInterface
```



only a subset!

Writing framework modules

When to use what kind of module

EDAnalyzer

- Reading data only
- Creating histograms
- the standard use case

EDProducer

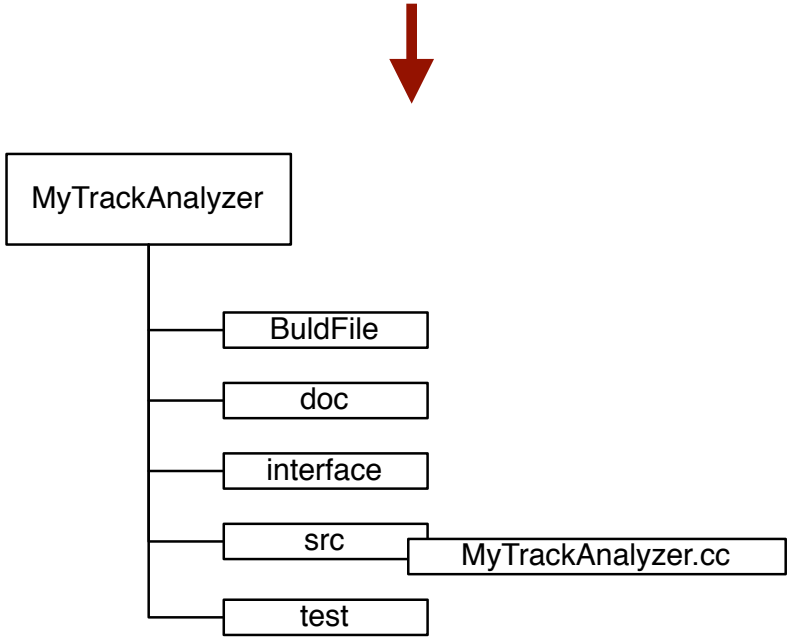
- You want to create new products
- You want to share your reconstruction code with others
- You want to make different algorithms plugable

EDFilter

- you want to know if an object could be produced
- you want to control the analysis flow or make skimming

Creating an EDAnalyzer

```
$ cd CMSSW_1_6_11/src
$ mkdir Tutorial
$ cd Tutorial
$ mkedanlzr -list
$ mkedanlzr -track MyTrackAnalyzer
```



EDAnalyzer source code

```
private:
   void beginJob( const edm::EventSetup& );
   void analyze( const edm::Event&, const edm::EventSetup& );
   void endJob();
```

```
// ------------ method called to for each event  ------------
void
MyTracks::analyze(const edm::Event& iEvent, const edm::EventSetup& iSetup)
{
   using namespace edm;
   using reco::TrackCollection;

   Handle<TrackCollection> tracks;
   iEvent.getByLabel("modulelabel",tracks);

   for(TrackCollection::const_iterator itTrack = tracks->begin();
       itTrack != tracks->end(); ++itTrack)
   {
      int charge = itTrack->charge();
   }
```

```
DEFINE_FWK_MODULE(MyTracks);
```

Building the example analyzer I

```
$ cd MyTrackAnalyzer
$ scramv1 build
```

BuildFile

```
<use name=FWCore/Framework>
<use name=FWCore/PluginManager>
<use name=FWCore/ParameterSet>
<flags EDM_PLUGIN=1>
<use name=DataFormats/TrackReco>
<export>
   <lib name=TutorialMyTrackAnalyzer>
   <use name=FWCore/Framework>
   <use name=FWCore/PluginManager>
   <use name=FWCore/ParameterSet>
   <use name=DataFormats/TrackReco>
</export>
```

Building the example analyzer II

```
$ cd MyTrackAnalyzer
$ scramv1 build
```

```
Reading cached build data
Scanning src/Tutorial/MyTrackAnalyzer/BuildFile

Entering Package Tutorial/MyTrackAnalyzer
Entering library rule at Tutorial/MyTrackAnalyzer
>> Compiling /build/hegner/CMSSW_1_6_12/src/Tutorial/MyTrackAnalyzer/src/MyTrackAnalyzer.cc
>> Building shared library tmp/slc4_ia32_gcc345/src/Tutorial/MyTrackAnalyzer/src/
TutorialMyTrackAnalyzer/libTutorialMyTrackAnalyzer.so
/usr/bin/ld: skipping incompatible /usr/lib64/libnsl.so when searching for -lnsl
...
/usr/bin/ld: skipping incompatible /usr/lib64/libc.a when searching for -lc
@@@@ Checking shared library for missing symbols: libTutorialMyTrackAnalyzer.so
/usr/bin/ld: skipping incompatible /usr/lib64/libm.so when searching for -lm
...
/usr/bin/ld: skipping incompatible /usr/lib64/libc.a when searching for -lc
@@@@ ----> OK, shared library FULLY-BOUND (no missing symbols): libTutorialMyTrackAnalyzer.so
@@@@ Checking shared library load: libTutorialMyTrackAnalyzer.so
@@@@ ----> OK, shared library loaded successfully: libTutorialMyTrackAnalyzer.so
Leaving library rule at Tutorial/MyTrackAnalyzer
--- Registered EDM Plugin: TutorialMyTrackAnalyzer
Leaving Package Tutorial/MyTrackAnalyzer
>> Package MyTrackAnalyzer built
```

There are many small tools you can use...

```
# Inspect a configuration
 $ edmConfigEditor <configfile>

# Dump the provenance information
 $ edmProvDump <rootfile>

# Create a code skeleton
 $ mkskel <name>
 $ mkedanlzr <name> / mkedanlzr <-template> <name>
 $ mkedprod <name>
 $ mkedfltr <name>

# Translating symbols into human readable strings
 $ c++filt <symbol>

# Other useful tools
 $ edm*
```

Where to get information

- CMS Workbook:
  https://twiki.cern.ch/twiki/bin/view/CMS/WorkBook
- Reference manual:
  http://cmsdoc.cern.ch/cms/cpt/Software/html/General/gendoxy-doc.php
- WEBcvs:
  http://cmssw.cvs.cern.ch/cgi-bin/cmssw.cgi/CMSSW/?cvsroot=CMSSW
- LXR:
  http://cmslxr.fnal.gov/lxr/
- HyperNews:
  https://hypernews.cern.ch/HyperNews/CMS/

Questions?
Comments?
Need a break?