Event Weighting

In order to run over both signal and background events in some generic analysis where the signal and background events carry a different weight it is useful to write that weight to the event.

Adding a weight to the event

First check out the code for UtilAlgos which now includes a producer to add a simple value of type double to the event. This will store the weight.

```
cvs co -r V05-12-01 PhysicsTools/UtilAlgos
scramv1 b
```

To use the new module in your CMSSW pset, for example adding an event weight of 10.7 the following module definition may be used when running over the set of events that require this weight. It must be added to a valid path.

```
module weight = DoubleProducer {
   double value = 10.7
}
```

Accessing the event weight

Add the following to your analyze method

```
edm::Handle<double> weightHandle;
event.getByLabel("weight", weightHandle);
double *weight = 0;
weight = weightHandle.product();
```

Then

```
weight
```

is a pointer to the weighting for a given event. Note that if no weight is present in the event the getByLabel will fail. As there is no way of testing the presence of a product in the event before attempting to get it this is problematic. By handling the exception this can be dealt with, although it's not a good solution. Something like:

```
double weight = 1.0;

try
{
   edm::Handle<double> weightHandle;
   event.getByLabel("weight", weightHandle);
   const double *weightPtr = weightHandle.product();
   weight = *weightPtr;
} catch(cms::Exception &ex) {
   edm::LogInfo("MyAnalysis") << "INFO:  No weight in event - weight is 1.0";
}

std::cout << "the weight is ... " << weight << std::endl;
```

-- DaveEvans - 26 Jun 2007

This topic: Main > DaveEvansEventWeighting
Topic revision: r3 - 2007-06-26 - DaveEvans