Table of Contents

How to write an LCSim Driver

A driver is something which is executed by LCSim. While LCSim is looping over an event sample it calls the existing driver(s) and allows them to access the information stored in the LCIO event. Digitization, reconstruction and analysis code is written in the form of drivers.

Methods

The following functions get called by the framework during the event loop. In order to execute code during certain stages of the loop process you have to overwrite the respective method.

```
protected void process(EventHeader)
```

: This method gets called on every event. It is the core of each driver. It passes the event header of the current event which allows to access all information stored in the LCIO event.

```
protected void startOfData()
```

: This method gets called once before the first event.

```
protected void endOfData()
```

: This method gets called once after the last event.

```
protected void suspend()
```

: This method gets called when the event loop is interrupted.

```
protected void resume()
```

: This method gets called when the event loop is resumed.

```
protected void detectorChanged(Detector)
```

: This method gets called whenever the detector changes between two events and once before the first event. It passes the new detector and should be used to querry detector information required by the driver.

Example

This example driver writes out the number of mc particles in each event.

Show... Hide

```
// import the required classes
import java.util.List;
import org.lcsim.event.EventHeader;
import org.lcsim.event.MCParticle;
import org.lcsim.util.Driver;

// the class has to be derived from the driver class
public class MyDriver extends Driver {

  // constructor
  public MyDriver() {
  }

  // overwrite the process method
  protected void process(EventHeader event) {
    // Get the list of mc particles from the event
    List<MCParticle> mcParticles = event.getMCParticles();
    // Print out the number of mc particles
    System.out.println("Event " + event.getEventNumber() + " contains " + mcParticles.size() + "
  }
}
```

Driver for XML Steering

In order to use your driver from the xml steering file you have to obey some rules

- The class must have a constructor which takes no arguments
- Only *public* methods without a return, which take only a single argument and start with *set* are available from the xml steering file, i.e. _public void setMyParameter(double x)_

This topic: CLIC > LCSimDriver
Topic revision: r2 - 2010-07-09 - ChristianGrefe