

JManager

The template class `JManager` resides in the name space `JROOT` and constitutes an auxiliary class for the creating and bookkeeping of many `ROOT` histograms, graphs, etc. It simply extends the functionality of an `STL` `map`. It has a master `ROOT` histogram, graph or other `TObject` that can be defined at construction. It then creates slave objects on the fly of which the name is obtained by replacing the wild card character in the name of the master by the corresponding key. All objects can be written to file with a single call. The basic implementation could be as follows.

```
template<class JKey_t, class JValue_t>
class JManager:
  public public JPointer<JValue_t>, // master histogram
  public std::map<JKey_t, JValue_t*> // STD map
{
  JManager(JValue_t* master); // master histogram

  JValue_t* operator[](JKey_t key);

  void Write(const char* file_name); // save data to file

  ..
};
```

The usual `map` operator is customised. In this, the availability of the object associated with the given key is checked. If necessary, a new object is cloned from the master. The name of this clone is determined from the name of the master by replacing the wild card character by the key. Finally, a pointer to the object associated to that key is returned.

For example.

```
JManager<int, TH2D> H2(new TH2D("H2[%]", NULL, 10, -1.0, +1.0, 10, -1.0, +1.0));

H2[12345678]->Fill(x, y);
```

The default wild card is the character `'%'`; it can be redefined at construction of a `JManager`.

As specified by `STL` `map`, the usual less-than operator `<` should be defined for the key. In addition, to create a name for the cloned object, the standard output stream operator `std::ostream<<` should be defined.

```
JManager<int, TH1D> H1(new TH1D("M_%]", NULL, 100, 0.0, 100.0e6));

while (..) {
  ..
  for (JDAQSuperFrame::const_iterator hit = frame->begin(); hit != frame->end(); ++hit) {
    H1[frame-getModuleID()]->Fill(hit->getT());}
  }
}

H1.Write(example.root);
```