gallery - Feature #16546

Find all products by type

05/15/2017 05:15 PM - Nathaniel Tagg

Status:	Closed	**Start date:**	05/15/2017
Priority:	Low	**Due date:**	
Assignee:	Kyle Knoepfel	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.10.00	**Spent time:**	30.00 hours
Scope:	Internal	**SSI Package:**	gallery
Experiment:	MicroBooNE		

Description

This isn't exactly needed, since I can hack it from the user level, but it might be useful to others.

I want to be able to get all data products of type recob::Wire from the current file.

Here's how I can hack it with custom code:

```
template<typename T>
vector<std::pair<string,art::InputTag>>  findByType(TTree* fTree)
{
  vector<std::pair<string,art::InputTag>> retval;
  // Leaf names for std::vector<recob::Wire> get renamed
  // to the art version of "recob::Wires" by this.
  std::string pattern = art::TypeID(typeid(T)).friendlyClassName();
  std::cout << "Looking for leaf of type " << pattern << "_label_instance_process." << endl;
  TObjArray* list = fTree->GetListOfBranches();

  // Look through every branch name.

  // We're looking for:
  // OBJ_LABEL_INSTANCE_PROCESSNAME.obj_
  // Where OBJ is something like "recob::Wires"
  // Where LABEL is something like "pandora"
  // INSTANCE is usually (always?) blank
  // Where PROCESSNAME is something like "McRecoAprStage1"

  for(int i=0;i<list->GetEntriesFast();i++) {
    TObject* o = list->At(i);
    TBranch* br = dynamic_cast<TBranch*>(o);
    std::string found = br->GetName();

    // Does it start with our object type?
    if(found.find(pattern)!=0) continue;
    // std::cout << "--Candidate: " << found << std::endl;

    // Does it end with '.'?
    string::size_type p3 = found.find_last_of('.');
    if(p3!=found.length()-1 || p3==0) continue;
    // std::cout << "p3 " << found.substr(p3) << std::endl;

    // Tokenize by underscore.
    string::size_type p2 = found.rfind("_",p3-1);
    if(p2==string::npos || p2==0) continue;
    // std::cout << "p2 " << found.substr(p2) << std::endl;

    string::size_type p1 = found.rfind("_",p2-1);
    if(p1==string::npos || p1 ==0 ) continue;
    // std::cout << "p1 " << found.substr(p1) << std::endl;

    string::size_type p0 = found.rfind("_",p1-1);
```

```
    if(p0==string::npos || p0==0) continue;
    // std::cout << "p0 " << found.substr(p0) << std::endl;

    art::InputTag tag(found.substr(p0+1,p1-p0-1),  found.substr(p1+1,p2-p1-1), found.substr(p2+1
,p3-p2-1));
    retval.push_back( make_pair( found, tag ));
  }
  return retval;

}
```

This code simply parses the Event tree branch list to find matching objects, and parses out the label/instance/process handles to create art::InputTags as a list. Then the calling code can do something like the following:

```
typedef vector<recob::Wire> thing_t;
for(auto item: findByType<thing_t>(ev.getTTree())) {
  cout<< "Branch: " << item.first << std::endl;
  cout<< "Tag   : " << item.second << std::endl;
  gallery::Handle< thing_t > handle;
  ev.getByLabel(item.second,handle);
  if(handle.isValid())
    std::cout << "Found " << handle->size() << " wires." << std::endl;
}
```

Looking at the ART source code, it would appear that this sort of parsing should be robust, since underscores are otherwise not allowed in InputTag elements.

History

#1 - 05/15/2017 05:29 PM - Nathaniel Tagg

Sorry, here's the top code in easier-to-read fashion:

```
template<typename T>
vector<std::pair<string,art::InputTag>>  findByType(TTree* fTree)
{
  vector<std::pair<string,art::InputTag>> retval;
  // Leaf names for std::vector<recob::Wire> get renamed
  // to the art version of "recob::Wires" by this.
  std::string pattern = art::TypeID(typeid(T)).friendlyClassName();
  std::cout << "Looking for leaf of type " << pattern << "_label_instance_process." << endl;
  TObjArray* list = fTree->GetListOfBranches();

  // Look through every branch name.

  // We're looking for:
  // OBJ_LABEL_INSTANCE_PROCESSNAME.obj_
  // Where OBJ is something like "recob::Wires"
  // Where LABEL is something like "pandora"
  // INSTANCE is usually (always?) blank
  // Where PROCESSNAME is something like "McRecoAprStage1"

  for(int i=0;i<list->GetEntriesFast();i++) {
    TObject* o = list->At(i);
    TBranch* br = dynamic_cast<TBranch*>(o);
    std::string found = br->GetName();

    // Does it start with our object type?
    if(found.find(pattern)!=0) continue;

    // Does it end with '.'?
    string::size_type p3 = found.find_last_of('.');
    if(p3!=found.length()-1 || p3==0) continue;

    // Tokenize by underscore.
    string::size_type p2 = found.rfind("_",p3-1);
    if(p2==string::npos || p2==0) continue;

    string::size_type p1 = found.rfind("_",p2-1);
    if(p1==string::npos || p1 ==0 ) continue;
```

```
    string::size_type p0 = found.rfind("_",p1-1);
    if(p0==string::npos || p0==0) continue;

    art::InputTag tag(found.substr(p0+1,p1-p0-1),  found.substr(p1+1,p2-p1-1), found.substr(p2+1,p3-p2-1));
    retval.push_back( make_pair( found, tag ));
  }
  return retval;

}
```

#2 - 05/17/2017 08:06 AM - Kyle Knoepfel

- Description updated

#3 - 05/22/2017 11:26 AM - Kyle Knoepfel

- Status changed from New to Feedback

Is a facility that matches the semantics of art::Event::getManyByType (in the art package) the one that is desired? If so, we would match the gallery interface to the corresponding interface in art.

#4 - 06/12/2017 11:45 AM - Kyle Knoepfel

From Nathaniel:

> First, I thought getManyByType was too hard, since I requested it about a year ago by email and didnt' get any response...? Looking at the source code, it wouldn't be easy to hack.
>
> Second, getManyByType doesn't do exactly what I want: I not only want the data products, but also the InputTag information, so I can output to the user. Maybe ART has that semantic in there somewhere.
>
> But you could easily wrap my suggestion to also give the same result as getManyByType.. that would work fine for most people.

#5 - 06/12/2017 11:58 AM - Kyle Knoepfel

> First, I thought getManyByType was too hard, since I requested it about a year ago by email and didnt' get any response...? Looking at the source code, it wouldn't be easy to hack.

We are sorry the email request was lost. Please submit future requests through the issues tracker system so they do not get lost.

> Second, getManyByType doesn't do exactly what I want: I not only want the data products, but also the InputTag information, so I can output to the user. Maybe ART has that semantic in there somewhere.

In order to keep the reading speed of gallery as fast as possible, Provenance information is not available. This is the information required to provide reliable InputTag construction.

> But you could easily wrap my suggestion to also give the same result as getManyByType.. that would work fine for most people.

Although we are able to provide the gallery::Event::getManyByType facility, unlike the art::Handle, the gallery::Handle does not provide provenance information. Given the lack of access to provenance information, is adding this facility to gallery still of interest?

#6 - 06/13/2017 03:53 PM - Nathaniel Tagg

Yes.

Here's the use case: an event viewer or simply a file cataloger. You know the file might contain Hits, but you want to list all the possible Hit lists the file contains so that the user can select the one they want (or put the correct name into an ART job so they can find them).

Again, simply adding my solution as a function in gallery::Event might be good enough for most users, and this is low priority for me (since you nicely provide a pointer to the raw Tree for me to operate on).

#7 - 06/19/2017 11:26 AM - Christopher Green

- Status changed from Feedback to Assigned

- Assignee set to Kyle Knoepfel

- SSI Package gallery added

#8 - 05/24/2018 02:10 PM - Nathaniel Tagg

No update for 11 months?

#9 - 05/30/2018 07:30 PM - Kyle Knoepfel

See #16547-7. Looks like I dropped the ball on this one--I probably should have put this back in "Accepted". As discussed in the aforementioned reference, we should have the ability to look at our backlog of issues in the next week or so.

#10 - 06/11/2018 10:26 AM - Kyle Knoepfel

- Target version set to 1.10.00

We intend to release *gallery*1.10.00 on the timescale of 1 or 2 weeks.

#11 - 06/14/2018 11:13 AM - Kyle Knoepfel

- Status changed from Assigned to Resolved

- % Done changed from 0 to 100

This feature has been implemented with commits gallery:66cc1ef and gallery:a332366. Some significant restructuring of *gallery* was necessary to provide the feature. The salient points are:

- getManyByType has been implemented with similar interface to *art*'s getManyByType
- The id() accessor has been added to the gallery::Handle and gallery::ValidHandle class templates, enabling product description retrieval.

Example of its use:

```
std::vector<gallery::Handle<MyProduct>> hs;
e.getManyByType(hs);

for (auto const& h : hs) {
  // Handles retrieved via 'getManyByType' are guaranteed to be valid
  assert(h.isValid());

  // Do something with the product
  auto const& prod = *h;

  // Where did the product come from?
  art::ProductID const id{h->id()};
  std::cout << e.getProductDescription(id).inputTag() << '\n';
}
```

#12 - 06/14/2018 12:16 PM - Kyle Knoepfel

- Status changed from Resolved to Closed