Subject: Basic library interface and separation problem
Posted by janwilmans on Mon, 08 Jun 2009 13:51:40 GMT
View Forum Message <> Reply to Message

Most of my projects aren't very well designed, when I even bother to separate different functionality into different libraries, I often use internal knowledge of the underlying library, even without thinking about it.

For my most recent project, I wanted to have proper separation and I ran into the following issue:

LibA
|
LibB
|
App

suppose we have library 'LibA' and 'LibB', LibB uses LibA and exposes part of LibA's functionality to an application using LibB.

LibA contains class A,

```
class A {
  enum a_attrs { foo, bar, foobar }
  static void setAttr(a_attris an_attr);
}
```

LibB contains class B,

```
class B {
  static void setAttr(a_attris an_attr)
  {
    A::setAttr(an_attr);
  }
}
```

This is a very bad solution for me, because I want to be able to replace LibA with LibA2 later, _and_ I don't want the App to even be aware of the existence of LibA.

Also, as it is now the App will include the headers of class A, indirectly, because App will include the header of class B.

So now, App has a source code level dependency on LibA .

One way to solve this would be to give class B it's own enum

```
class B {
  enum b_attrs { b_foo, b_bar, b_foobar }
  static void setAttr(b_attrs an_attr);
```

```
}

void B::setAttr(b_attrs an_attr)
{
  switch(an_attr)
  {
  case b_foo:
    A::setAttr(foo);
    break;
  case b_bar:
    A::setAttr(bar);
    break;
  case b_foobar:
    A::setAttr(foobar);
    break;
  default:
    break;
  }
}
```

But when I add an attribute to class A, in would also have to edit class B to add the attribute there as well.

Any thought on how to solve this neatly ?

Greetings,

Jan