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

Subject: Re: Basic library interface and separation problem
Posted by koldo on Mon, 08 Jun 2009 20:57:39 GMT

View Forum Message <> Reply to Message

Hello skyhawk

Sorry, with so many A, and B, and foo, and bar, ..., I am lost!.

For me, when I see that I have two classes that share things I have three options:
- Redesign them dividing the functionalities so that finally I have different classes
- Make a common class with common funcionalities and derive both from this common class.

- Join them in one class.

You have to know how to cut and remix the classes. Or put a more defined sample and we can tell you our opinion.

Best regards
Koldo

Subject: Re: Basic library interface and separation problem
Posted by Didier on Mon, 08 Jun 2009 22:25:20 GMT
View Forum Message <> Reply to Message

Usualy what is used for this is: "dependancy inversion"
In other words: make an interface (it will be the API) and make both librairies depend on it (solution 2 of koldo).

The result is B does not depend on lib A although it uses it !
You can now freely change lib A or B, as long as you keep the API stable.

This is what APIs are made for

Subject: Re: Basic library interface and separation problem
Posted by janwilmans on Tue, 09 Jun 2009 06:22:33 GMT
View Forum Message <> Reply to Message

Koldo, sorry for the confusion, what I was looking was that:

When two classes need to share more then just basetypes, in my opinion you have two options:

- make one class depend on the other
- create a third class that serves as 'common' API for both.

I my case, because the two class are in separate libraries and I do not want source code dependencies between them; I've done what Didier and you suggested, created a third library that is shared/depended on by the other two which contains a class that serves as common API for both libraries.

Anyway, thanks for your swift responses!

Jan