

The pImpl Idiom Solutions

The plmpl Idiom

- What is the plmpl idiom?
 - The plmpl idiom implements the Handle-Body pattern
 - The Handle class has a private data member which is a pointer to a Body object
 - This is a "pointer to the implementation" (plmpl)
 - A Body object is allocated in the Handle's constructor and released in the Destructor
 - The member functions of the Handle forward their arguments to the corresponding member functions of the Body

The plmpl Idiom

- How does the plmpl idiom help with the separation problem?
 - Since Handle has a pointer to a Body, it only needs a declaration of Body, not the entire class definition
 - It therefore does not need to include Body's header
 - Since the clients only interact with a Handle, they do not need to include Body's header either
 - Clients will not be aware of any changes in the Body implementation (provided the Handle interface is unchanged)
 - Clients do not need to recompile when the Body implementation changes

Pros and cons

- What other advantages does the plmpl idiom have when compiling and distributing large applications?
 - Since the Body implementation is separate from the rest of the program, it can be compiled into a DLL/shared object
 - (A DLL/shared object is loaded by the program when it starts executing)
 - This means that changes to the Body implementation only require recompiling the DLL
 - If there is a change to the Body interface which affects the Handle's implementation, but not the Handle's interface, client code does not need to be re-compiled (only re-linked)

Pros and cons

- What disadvantages does it have?
 - Requires an extra memory allocation
 - Member functions calls require a pointer dereference
 - Adds complexity

plmpl with unique_ptr

- Write a simple program that uses the plmpl idiom with unique_ptr
- Your program should organize the code in the usual way (class definition in header file, member function definitions in source file)