Subject: Re: true dynamic dispatching with Upp?
Posted by Lance on Wed, 05 Dec 2012 03:24:30 GMT
View Forum Message <> Reply to Message

It depends on whether the exact type of the Elements that you(or more precisely, your code) receives can be determined at compile time or not. If the answer is yes, template specialization is the fastest way to go.

```
template <class T>
struct ElementEditorFinder;

template <>
struct ElementEditorFinder<ElmentA>
{
    typedef ElementAEditor Editor;
};

template <>
struct ElementEditorFinder<ElmentB>
{
    typedef ElementBEditor Editor;
};
```

But most likely the answer is no. In that case, Dider's solution is insufficient. (Sorry Dider, in no offence, and again, I might be wrong as I always do . If you code is given a Elemnt * which you don't know the exact class name, how are you going to find the correct ElementHelperBase class from it? The time and path taken would be quite similar to when you find the Editor class directly (without using the HelperBase class).

The reason why it's slow is because you do it in a sequential way, plus repeated dynamic_cast might also be costly. You may work around this by using a map or sorted vector or some other facilities support(log(n) time complexity). If the Elements class hierachy happens to provide a distinct integral ID of each Elements class, by all means, use it as the key, otherwise, use the typeid string. For the value field, you cannot use the Editor but you can use a function pointer to a function that will generate a correct matching Editor for the value field.

It takes some extra resources to build the map (or sorted Vector), but if there are really a lot of Elements classes, it might worth the effort.

HTH,
Lance