Visitor and Proxy

10/28/2009

Opening Discussion

- Do you have any questions about the reading for today?

Visitor

- This pattern provides a way to perform an operation on all of the elements of some structure. This allows you to change the operation without having to alter the code for the structure itself.
- To do this we define a Visitor interface with a method that is called when an object is “visited” that method also accepts the object being visited as an argument.
- The ConcreteVisitor can collect information, or might be like a simple functor.

Example

- GoF uses an example of a compiler where the code is compiled to a meta-format and then we pass through that format several times doing different things.
- Without visitor, we need a different method for each type of pass we want to do. That adds a new method to every type in our meta-format. With visitor we just create a new ConcreteVisitor.
- More familiar to you would be having a binary tree and wanting to do different things to the elements.

Benefits and Drawbacks

- The primary benefit is that it is very easy to add new operations. It is also easy to change that functionality for a single visitor because it sits in a single visitor class.
- This also keeps things related to different operations separated into different visitors.
- When you want to add a new type to the structure, that can be hard. This is because different visitors might handle each subtype differently.

More Benefits and Drawbacks

- Visitors can work across class hierarchies. An iterator can't do this. The visitor can also be more efficient than an iterator in some cases.
- The visitor can accumulate state where data would have to be passed through if the methods were part of the objects.
- Visitors often require you to break encapsulation in some way.

Proxy

- With this pattern you provide a placeholder or surrogate for a given object that gives you remote/indirect access to it.
- The primary use of this is when the creation of certain objects is expensive, we don't want to create them unless they are needed and only when they are needed. Instead we create a proxy object that can answer simple questions about the full object and only instantiates the full object when required.

Example

- GoF uses the example of an image in a document. These can be expensive to load and store so the proxy only gets enough information for the size, not the full raster. The full raster is created and remembered only when it is viewed.
- This pattern is very similar in many ways to how dynamic loading can happen in the project. We keep a proxy object for things that aren't currently viewed and those objects load when needed.
- RMI also works a a Proxy.
- In C++ the auto_ptr class is a proxy for pointers. Similar things can be done that allow reference counting.

Benefits and Drawbacks

- The extra level of indirection for a Proxy enables numerous types of enhancements depending on the type of proxy object.
  - Remote proxy
  - Virtual proxy
  - Protection proxy
  - Smart reference
- A copy-on-write proxy allows sharing of large objects as long as they aren't changed. It includes reference counting. The large object is only copied when modified.

Progress Presentations

- Does anyone want to present anything this week for progress on the project?
- It is critical to understand that sometimes you just have to pick a way to do something and do it.