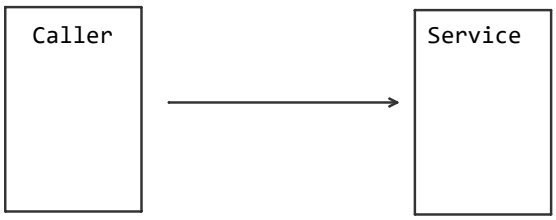
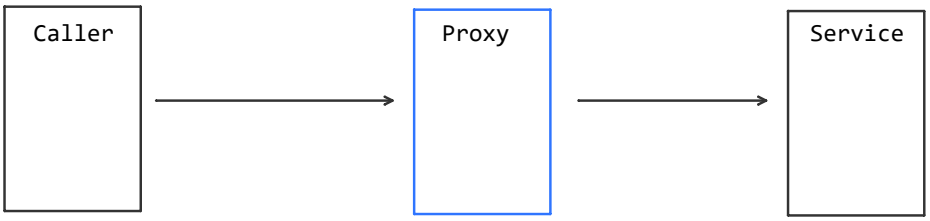
Proxy Pattern

06 June 2021 07:54 PM

The Proxy Pattern provides a surrogate or placeholder for another object to control access to it.

Instead of calling the service directly the proxy is called which then calls the service

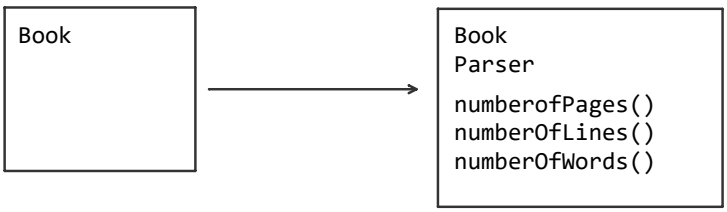




Proxy can be of 3 types:

1. Remote Proxy - for accessing remote resources(different server, different namespace, different project)
2. Virtual Proxy - controls access to a resource that is expensive to create(caching)
3. Protection Proxy - controls access to a resource based on access rights (Access management)

Example




```
Book b = new Book()
BookParser bp = new BookParser(b)

bp.numberOfPages()
bp.numberOfLines()
bp.numberOfWords()
```

Suppose we want a book to be parsed that's encrypted, etc.
And it provides functions to get the number of pages, words, lines.

Now the situation is, when we instantiate the BookParser with the book. It starts calculating…and utilizes the memory and if in some cases we don't require the functions the resources are wasted and the performance is decreased

Possible solution is to make the BookParser lazy that starts calculating when the functions are called.

BUT, if this book parser is a network service or in some other namespace(which we can't edit). Then the Proxy pattern comes to rescue.

We can make the proxy pattern lazy solving the performance issues.

General UML Diagram