Hochschule für angewandte Wissenschaften
Fachhochschule Würzburg-Schweinfurt
Fakultät Informatik und Wirtschaftsinformatik

Design Patterns

Proxy

Christoph Okelmann

5. Januar 2013

Inhaltsverzeichnis

1 Einführung

Proxy ist ein englisches Wort und heißt übersetzt Stellvertreter. Die lateinische Abstammung kommt vom Wort „proximus“, was der/die/das Näheste heißt und somit die Wortverwandschaft zu „Approximation“ zeigt.
Meistens verstehen wir unter einem Proxy einen Server der vorzugsweise vor eine Datenbank geschaltet ist (siehe Abb. 1.1), damit Daten aufbereitet, gefiltert oder gleichzeitige Zugriffe zwischengespeichert werden können, so zum Beispiel beim IMAP-Proxy, der E-Mail Services bereitstellt.
Als Design Pattern ist ein Proxy ein Stellvertreter für einen Dienst, der über Ausführbarkeit oder Berechtigung entscheidet und danach den eigentlichen Dienst aufruft, bzw. wenn der jeweilige Dienst noch beschäftigt ist eine Anfragewarteschlange erzeugt. Ein typisches Beispiel für die Verwendung eines Proxy als Design Pattern wäre die Benutzerkontenverwaltung von Windows, bzw. besser gesagt, dass Rechtesystem das dahintersteckt (siehe Abb. 1.2).

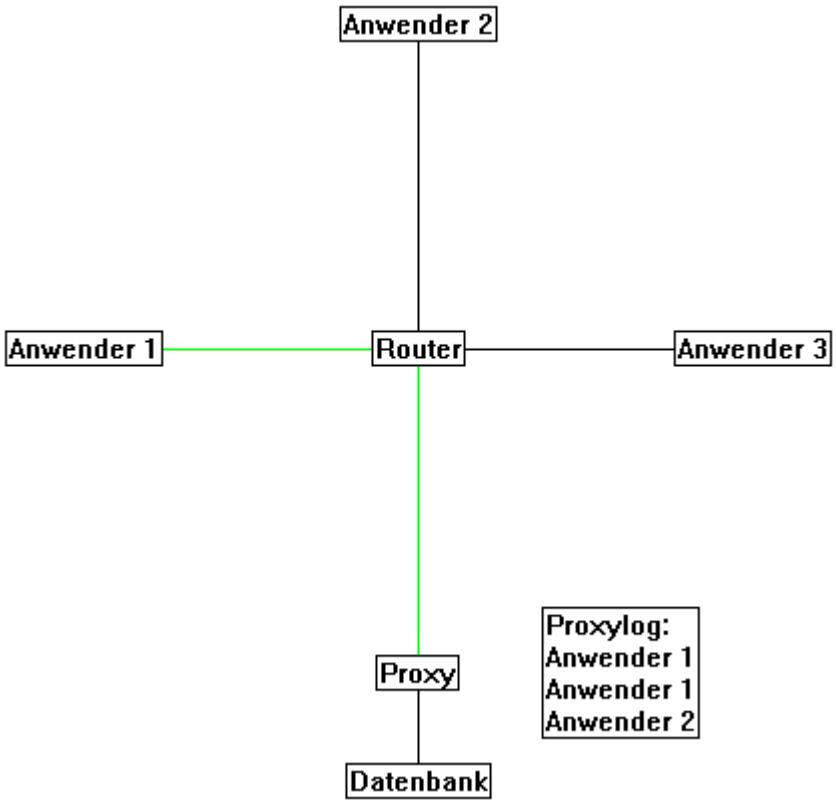

Abbildung 1.1: Datenbankproxy

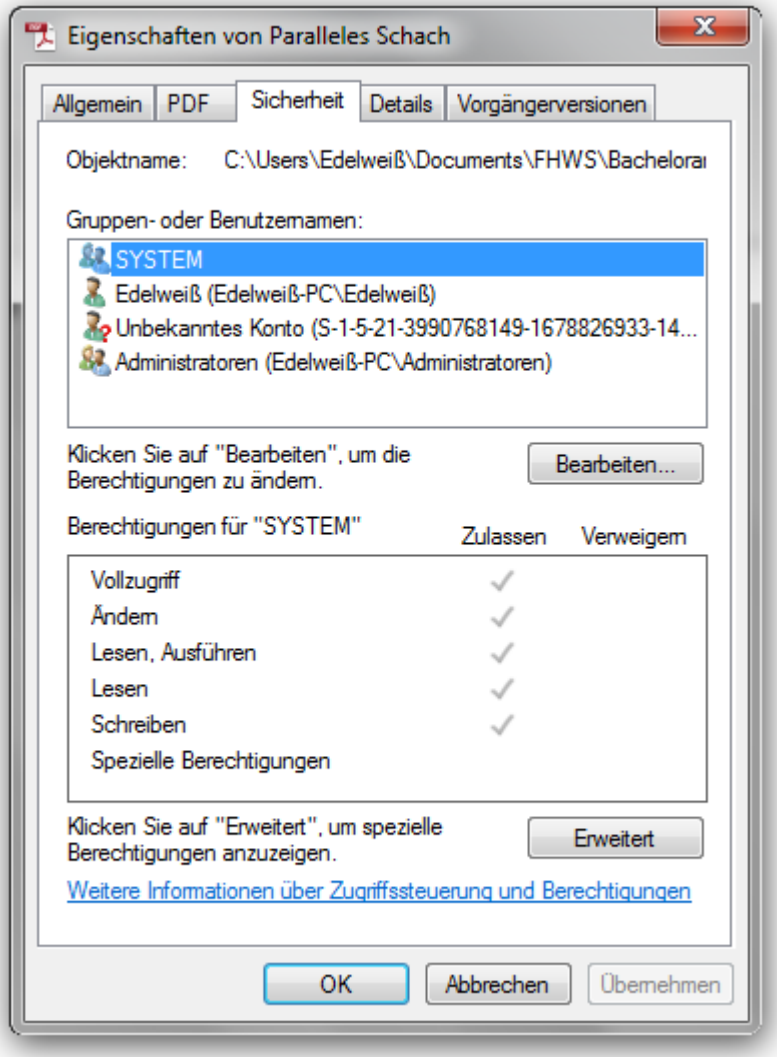

Abbildung 1.2: Benutzerkontenverwaltung

2 Historische Entwicklung

Die Design Patterns wurden ursprünglich vom römischen Architekt Vitruv entwickelt.

Abbildung 2.1: Der römische Architekt Vitruv

Natürlich nicht unter diesem Namen usw., aber die Grundzüge waren da als Entwurfsmuster für Gebäude, bei denen die zukünftigen Bewohner sehr gut mitbestimmen sollten, wie das Gebäude auszusehen hat. Natürlich haben diese keine architektonische Erfahrung und sind auf möglichst einfache und umfassende Informationen des Architekten angewiesen. Demzufolge kann man sich klar vorstellen wie dieser Prozess damals ungefähr auszusehen hatte. In der Postmoderne nahm man diese Idee, die in der Architektur seit damals als Tradition weitergeführt wurde, zum Entwurf für Graphische Benutzeroberflächen auf. Diese Vorgehensweise wird auch, wie jedem Programmierer bekannt intensiv weiterverwendet. So zu sehen in Visual Studio im sogenanntem Werkzeugmanager, da die sogennanten „Controls" immer genau gleich strukturiert sind.
Später kam die Idee hinzu, da man bemerkte, dass viele nichtgraphische Programmierelemente auch immer gleichgeartet sind, hierfür eine ähnliche Sammlung anzulegen. Warum diese nicht als Objektorientierte Datenstrukturen vordefiniert sind mag daran liegen, dass diese grundsätzlich eher einfach gehalten sind oder eher individuell anpassbar sein sollten. Deswegen existiert auch bis heute meistens keine vorimplementierte Version, die wie bei der Zusammenstellung von graphischen Benutzeroberflächen, direkt weiterverwendet werden könnte.
Im Zuge der Sammlung fiel auf, dass man hier bisher drei Kategorien unterscheiden kann. Da ein besonderes Merkmal eines Teiles der Design Patterns die Anfertigung eines neuen Objektes ist nannte man diese die Erzeugungsmuster. Alle anderen wurden dann noch in zwei weitere Kategorien unterteilt, die sog. Strukturmuster, die man wohl eher in Verbindung bringt mit der Kategorie der Sammlung aller Design Patterns bis man diese zuordnen konnte, also eher einfache Nichterzeugungsmuster und die Verhaltensmuster, für die man erkannte, dass diese eher komplex wurden oder im Zuge der Entwicklung von höheren Pro-

grammen dann erst entstanden und bereits komplexe Datenstrukturen und Eventhandling beeinhalten bwz. dafür entwickelt wurden.

3 Proxy als Design Pattern

Wie wir innerhalb Abb. 3.1 sehen können handelt es sich bei einem Proxy um ein Codemuster, dem ein Dienst zugewiesen wird. Innherhalb des Proxy wird bei Aufruf entschieden, ob und wann der Dienst ausgeführt werden soll.

Listing 3.1: Beispiel

```
class Proxy
{
    public Dienst dienst;
    public int usernummer;

    public void Execute(object data)
    {
        if (usernummer > 1000)
            System.Console.WriteLine("Zugriff␣verweigert");
        else
            dienst.Execute(data);
    }
}
```

Innerhalb des Codebeispiels sehen wir einen sog. Schutzproxy, der bei Aufruf anhand von Zugriffsrechten entscheidet, ob der Dienst ausgeführt wird. Außerdem gibt es noch folgende andere Arten von Proxys:

1. Remote-Proxy
2. Virtueller Stellvertreter
3. Schutzproxy
4. Smart Reference

Natürlich sind das nur die am häufigst vorkommenden Typen von Proxys. Ein Beispiel für eine weitere Variation ist eine Kombination mit dem Flyweight Design Pattern. Das Flyweight Design Pattern wird zum Beispiel bei längeren Texten verwendet, also bei einer Textlänge, die dazu führen würde, dass die Ladezeiten oder der Speicherbedarf zu groß wäre, um den gesamten Text einschließlich Satzinformationen zu laden, auf jeden Fall, wenn man für jedes Zeichen das Format usw... angeben würde. Dann wird auf ganz natürliche Art und Weise die Information unredundant abgelegt und deswegen unterschiedliche Datentypen und bei Darstellung von Texten zum Beispiel der Bereich in dem ein Zeichensatz aktiv ist usw... zur Vereinfachung mitverwendet.

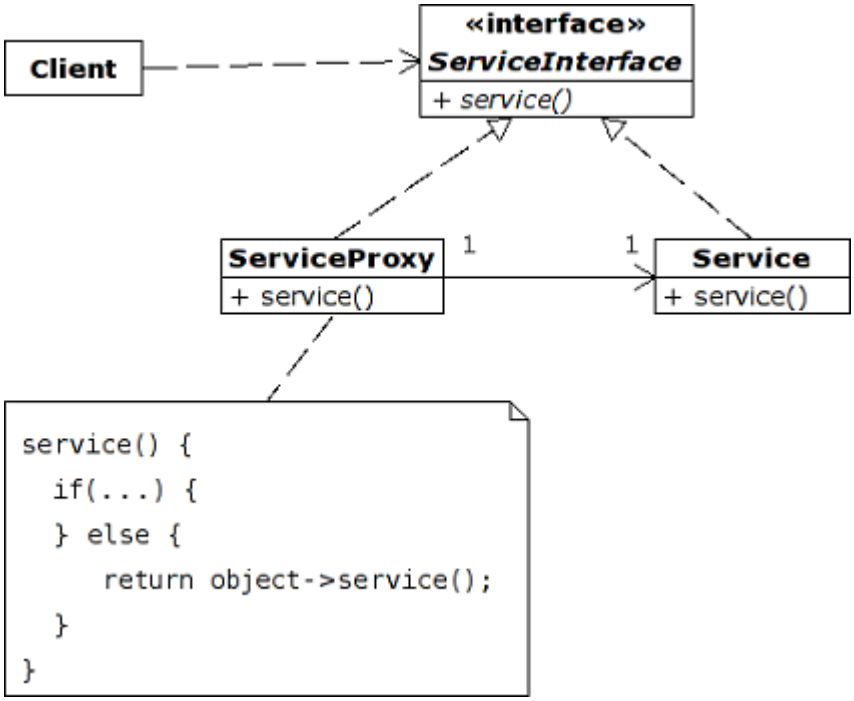

Abbildung 3.1: Proxy

4 Einordnung zu Strukturmustern

Innerhalb der Design Patterns gibt es drei große Gruppen. Die Erzeugungsmuster (Factory usw...), Strukturmuster (eher eindimensional) und die Verhaltensmuster, die von den Strukturmustern abgeleitet, aber komplexer sind.
Wie bereits oben erwähnt sind die Strukturmuster eher einfachere Design Patterns, die keine Instanzen erzeugen können. Da der Proxypattern sehr einfach ist, da dieser in der Hauptmethode meist nur eine Bedingung abfragen muss und dann den Dienst ausführt oder eher wartet bis der Dienst auszuführen ist (mit Aufrufwarteschlange/ -queue) bzw. gleichzeitig mit dem eigentlichen Dienst eine Reihe von begleitenden sehr einfachen Diensten, wie zum Beispiel zählen der Aufrufe usw..., durchführt und nicht hauptsächlich zur Verwendung für komplexe Datenstrukturen und Eventhandling ausgelegt worden ist, zählt der Proxypattern zu den Strukturmustern.
Der Proxypattern wird zum Beispiel hauptsächlich als Sicherheitsproxy (siehe Windows-Dateirechtesystem) verwendet. Für diese Datenstruktur gibt es keine vorimplementierte Version als Klasse einer bereits existierenden Library, obwohl eine Version als Datenstruktur mit Übergabe einer Funktion zur Entscheidung der Durchführung, bzw. Aufruf von Nebendiensten oder Warten bis zur Auslastungsflaute und mit Übergabe des eigentlichen Dienstes denkbar wäre.

Erzeugungsmuster (Creational Design Patterns)	Strukturmuster (Structural Design Patterns)	Verhaltensmuster (Behavioral Design Patterns)
Abstrakte Fabrik (abstract factory pattern)	Adapter (adapter pattern)	Beobachter (observer pattern)
Einzelstück (singleton pattern)	Brücke (bridge pattern)	Besucher (visitor pattern)
Erbauer (builder pattern)	Dekorierer (decorator pattern)	Interceptor (interceptor pattern)
Fabrikmethode (factory method pattern)	Fassade (façade pattern)	Interpreter (interpreter pattern)
Multiton	Fliegengewicht (flyweight pattern)	Iterator (iterator pattern)
Prototyp (prototype pattern)	Kompositum (composite pattern)	Kommando (command pattern)
	Stellvertreter (proxy pattern)	Memento (memento pattern)
		Nullobjekt (Null Object pattern)
		Schablonenmethode (template method pattern)
		Strategie (strategy pattern)
		Vermittler (mediator pattern)
		Zustand (state pattern)
		Zuständigkeitskette (chain of responsibility)

Abbildung 4.1: Kategorien von Design Patterns nach Wikipedia

5 Unterschiedliche Proxyarten

Die geläufigsten Proxyarten sind:

1. Remote-Proxy
2. Virtueller Proxy
3. Schutzproxy
4. Smart Reference

Der Remote-Proxy ist sehr ähnlich dem eigentlichem Proxyserver bzw. könnte es sein, dass der Proxyserver dieses Software Design Pattern verwendet, da es sich bei einem Remote-Proxy um eine in einem entfernten Teil des Netzwerkes vorhandene Ressource handelt, die lokal vom Proxy (Design Pattern) aus zugreifbar ist.

Der Virtuelle Proxy überprüft vor dem Ausführen des eigentlichen Dienstes die Auslastungslage auf dem System und wartet so lange bis eine geeignete Auslastungslage zur Ausührung des Dienstes gefunden ist oder vermutet wird.

Beim Schutzproxy handelt es sich um die Variante des Rechtesystems, wie zum Beispiel innerhalb von Windows beim Ausführen oder Verändern von Dateien, der die Zugriffsrechte verwaltet und überprüft und eine Durchführung des Dienstes gewährleistet oder eine Fehlermeldung ausgibt.

Die Smart Reference ist ein Proxy der vor dem Ausführen des eigentlichen Dienstes diverse kleinere Dienste, wie zum Beispiel das Zählen der Persistenzzugriffe, ermöglicht. Denkbar wäre eine Überprüfung der Daten innerhalb einer Datenbank nach Bedarf eines weiteren Backups, wenn mehr als 100 Datensätze verändert wurden, zum Beispiel bei längeren Artikellisten, die sich nicht oft ändern.

6 Abwegigere Proxyart

Eine abwegigere Proxyart ist eine Kombination mit dem Flyweight Design Pattern. Das Flyweight Design Pattern dient dazu zum Beispiel sehr lange Texte, bei denen es nicht nötig ist die gesamte Datei zu laden, trotzdem in einem gepacktem Format innerhalb des Speichers abzulegen, so dass nicht zu jedem Zeichen der Zeichensatz, Schriftgröße und Schriftsatz kombiniert vorliegen muss. Das heißt nach dem Laden jedes Zeichens mit zugehöriger Information innerhalb des Bereiches wird eine Normalisierung der Daten durchgeführt und die daraus gewonnenen Informationen mit Bereichsangaben, zum Beispiel: Seite 3-4 Latin Schriftgröße 6, möglichst kompakt abgelegt. Die Umnormalisierung ist teilweise notwendig, da es möglich ist das Bereichsangaben von Textsatzinformationen den geladenen Bereich überschreiten und trotzdem innerhalb des geladenen Bereichs eine Änderung vorliegen kann.
Die Proxyvariante, die ähnlich aufgebaut worden ist sieht vor ein großes Datenpaket einmalig im Speicher abzulegen und bei Bedarf von unterschiedlichen Stellen aus mittels der Proxys abrufen oder verändern zu können. Eine denkbare Variante wäre ein Operationslogbuch für jeden Proxy mitzuführen, das „gemeinsame Nenner" analysiert und bis dorthin jeweils einen gemeinsamen Datenblock mitführt und weitere Verarbeitungen jeweils undynamisch wiedererzeugt. Wie in Wikipedia beschrieben scheint es jedoch eher so, dass der Datenblock immer einmalig, gemeinsam und bei einem Operationsaufruf eines Proxys ab diesem Zeitpunkt für jeden Proxy geändert vorliegt.

7 Fazit

Die drei unterschiedlichen Kategorien von Design Patterns scheinen im Verlauf klar geworden zu sein: Erzeugungs-, Struktur- und Verhaltensmuster. Der historische Verlauf ist interessant und zeigt, dass diese Design Patterns meistens nicht als vordefinierte Klassen vorliegen (Ausnahme wäre zum Beispiel „Observer"), vielleicht weil man sich darauf geeinigt hat, dass diese Muster zu einfach sind um diese dem Programmierer als Lernübung vorzuenthalten oder weil die Informatik noch eine sehr junge Wissenschaft ist, deren theoretischer Teil noch nicht abgeschlossen ist oder noch Platz für eine Weiterentwicklung bereitstellen muss.