Proxy Design Pattern

Intent

- Provide a surrogate or placeholder for another object to control access to it.

- Use an extra level of indirection to support distributed, controlled, or intelligent access.

- Add a wrapper and delegation to protect the real component from undue complexity.

Problem

You need to support resource-hungry objects, and you do not want to instantiate such objects unless and until they are actually requested by the client.

Discussion

Design a surrogate, or proxy, object that: instantiates the real object the first time the client makes a request of the proxy, remembers the identity of this real object, and forwards the instigating request to this real object. Then all subsequent requests are simply forwarded directly to the encapsulated real object.

There are four common situations in which the Proxy pattern is applicable.

1. A virtual proxy is a placeholder for "expensive to create" objects. The real object is only created when a client first requests/accesses the object.

2. A remote proxy provides a local representative for an object that resides in a different address space. This is what the "stub" code in RPC and CORBA provides.

3. A protective proxy controls access to a sensitive master object. The "surrogate" object checks that the caller has the access permissions required prior to forwarding the request.

4. A smart proxy interposes additional actions when an object is accessed. Typical uses include:
   - Counting the number of references to the real object so that it can be freed automatically when there are no more references (aka smart pointer),
   - Loading a persistent object into memory when it's first referenced,
   - Checking that the real object is locked before it is accessed to ensure that no other object can change it.

Structure

By defining a Subject interface, the presence of the Proxy object standing in place of the RealSubject is transparent to the client.
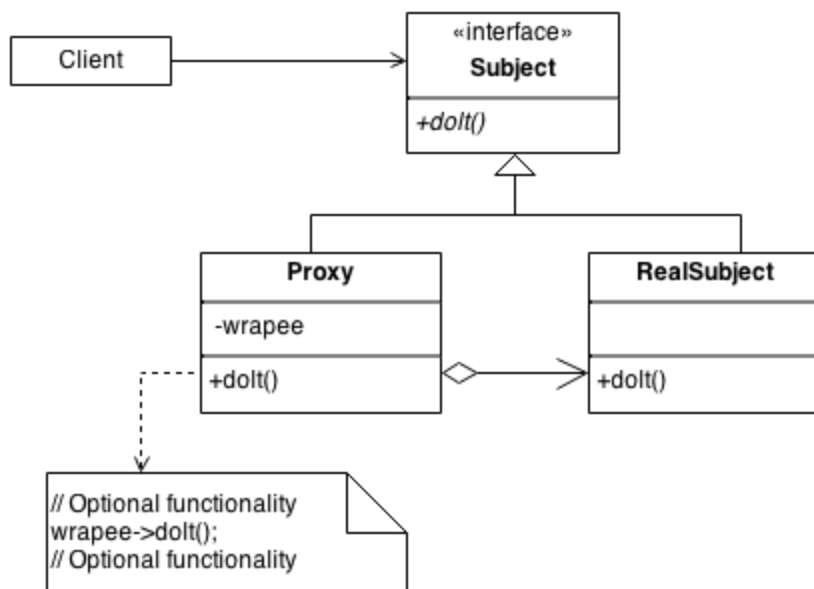
Example

The Proxy provides a surrogate or place holder to provide access to an object. A check or bank draft is a proxy for funds in an account. A check can be used in place of cash for making purchases and ultimately controls access to cash in the issuer's account.
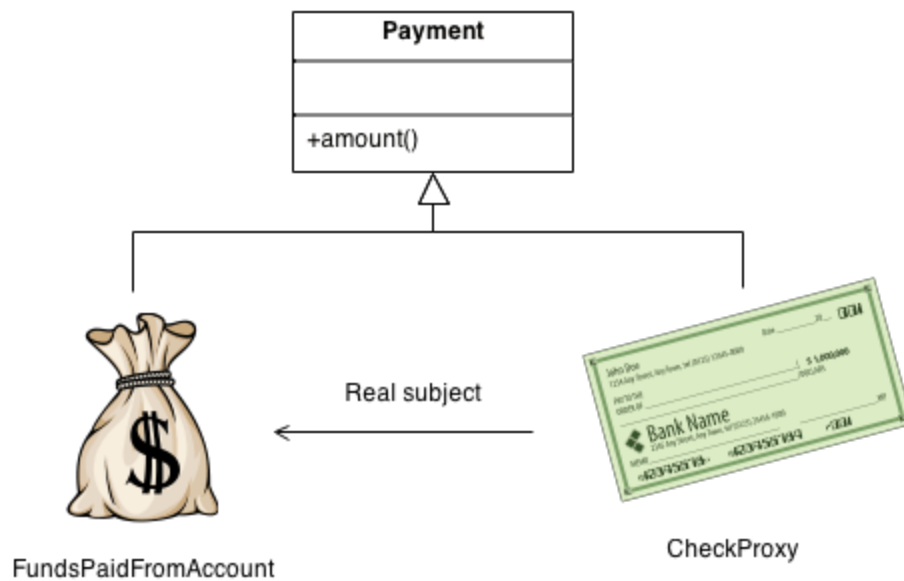
Check list

1. Identify the leverage or "aspect" that is best implemented as a wrapper or surrogate.

2. Define an interface that will make the proxy and the original component interchangeable.

3. Consider defining a Factory that can encapsulate the decision of whether a proxy or original object is desirable.

4. The wrapper class holds a pointer to the real class and implements the interface.

5. The pointer may be initialized at construction, or on first use.

6. Each wrapper method contributes its leverage, and delegates to the wrappee object.

Rules of thumb

- Adapter provides a different interface to its subject. Proxy provides the same interface. Decorator provides an enhanced interface.

- Decorator and Proxy have different purposes but similar structures. Both describe how to provide a level of indirection to another object, and the implementations keep a reference to the object to which they forward requests.

Support our free website and own the eBook!

- 22 design patterns and 8 principles explained in depth
- 406 well-structured, easy to read, jargon-free pages
- 228 clear and helpful illustrations and diagrams
- An archive with code examples in 4 languages
- All devices supported: EPUB/MOBI/PDF formats

📖 Learn more...

Code examples

Java	**Proxy in Java**		
C++	**Proxy in C++: Before and after**	**Proxy in C++**	**Proxy in C++**
PHP	**Proxy in PHP**		
Python	**Proxy in Python**		

⭐ More info, diagrams and examples of the Proxy design pattern you can find on our new partner resource Refactoring.Guru.

READ NEXT

Behavioral patterns →

RETURN

← Design Patterns

Private Class Data

My account

AntiPatterns

Forum

Refactoring

Contact us

UML

About us

Terms / Privacy policy