

Design Pattern Proxy - 1

Intento:

Definire un surrogato per un oggetto per controllare gli accessi ad esso.

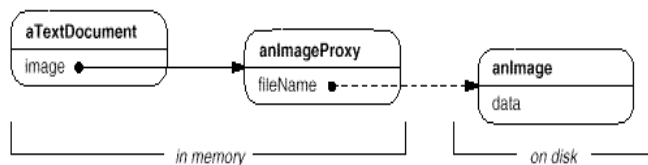
Es: in un editor all'apertura di un documento bisogna ridurre i tempi di creazione di oggetti che contengono immagini.

Problema:

Nascondere che l'immagine è creata solo quando serve per non complicare gli altri oggetti dell'editor

Soluzione:

Usare un oggetto Proxy che si frappone tra utilizzatori ed immagine. Il Proxy crea l'immagine solo quando l'editor invoca l'operazione `draw()`



Design Pattern Proxy - 2

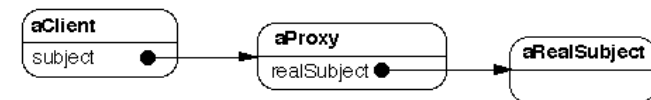
Componenti:

RealSubject è l'oggetto rappresentato dal Proxy

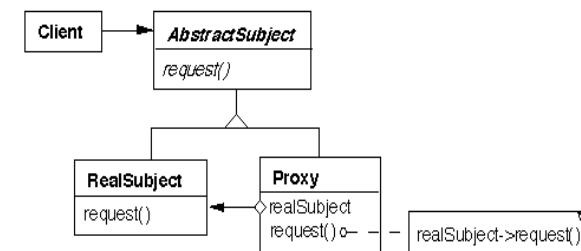
Proxy (1) contiene il riferimento che permette l'accesso al RealSubject; (2) fornisce l'interfaccia del RealSubject; (3) controlla l'accesso al RealSubject

- La variante **Remote Proxy** si occupa di mandare le richieste ad un host remoto dove il RealSubject risiede e di ricevere risposte dal remoto.

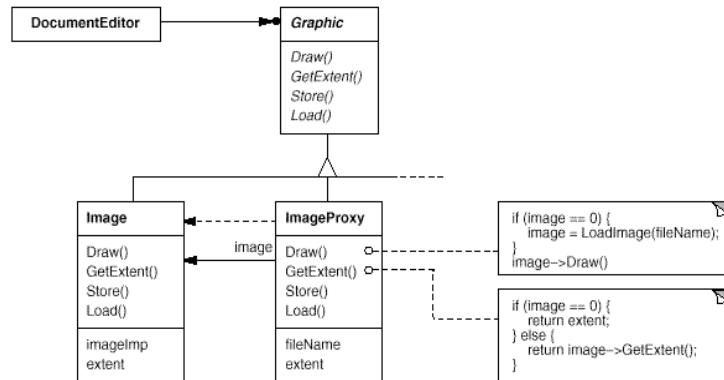
AbstractSubject definisce un'interfaccia per RealSubject e Proxy così che il Proxy può essere usato al posto di RealSubject



Struttura:



Design Pattern Proxy - 3



Conseguenze:

- Il Proxy introduce un'indirettezza.
- Nasconde il fatto che il `RealSubject` risiede in un host remoto
- Ottimizza la creazione di oggetti onerosi
- Permette di implementare politiche di protezione degli accessi (autenticazione e autorizzazione, oppure lock)
- Ottimizzazione copy-on-write

Usempi di utilizzo:

Il sistema operativo NeXTSTEP crea un Proxy la prima volta che si chiede un accesso ad un servizio remoto.

OMG-CORBA usa gli stub per accedere ai server ed all'ORB e gli skeleton per permettere all'ORB di mandare richieste al server. Orbix, una implementazione di OMG-CORBA, usa i Proxy.

Design Pattern Broker - 1

Intento:

Strutturare sistemi distribuiti in modo da disaccoppiare i componenti che interagiscono per fornire o usare un servizio tramite invocazioni remote. Ciascun componente si concentra su una specifica funzionalità.

Es: un sistema di informazioni per una città che deve eseguire in una WAN. I client che forniscono dati su hotel, ristoranti, etc. non conoscono chi sono e dove sono i server.

Problema:

L'ambiente è distribuito e possibilmente eterogeneo.

Ciascun componente che interagisce dovrebbe accedere a servizi forniti in remoto senza conoscere la posizione dei servizi.

I componenti possono essere scambiati, aggiunti o rimossi.

Gestione dei fault.

Soluzione:

Un componente (detto **Broker**) è responsabile per coordinare le comunicazioni tra client e server.

I server si registrano con il Broker e rendono disponibili i loro servizi ai client tramite una interfaccia resa nota.

I client accedono alle funzionalità dei server attraverso il Broker.

Il Broker si occupa di localizzare il server appropriato, di inviargli richieste e di trasmettere indietro i risultati (o le eccezioni).

Design Pattern Broker - 2

Componenti:

Client e Server

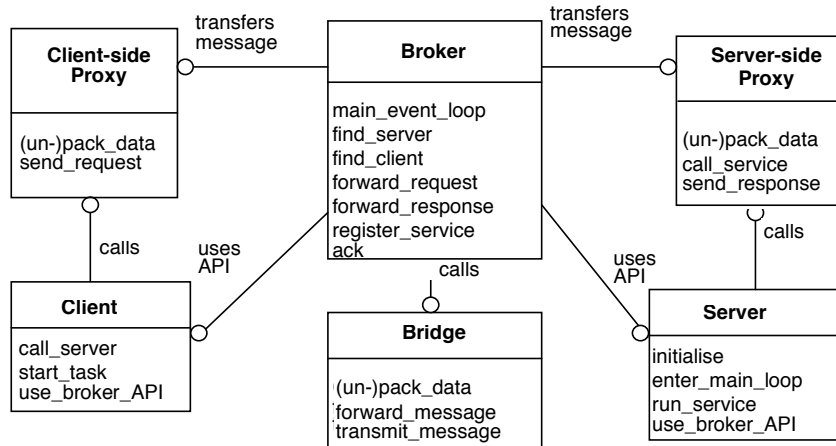
Client-Side Proxy conosce come dialogare con il Broker e media tra il Client ed il Broker.

Server-Side Proxy chiama i servizi dal lato server e media tra il Server ed il Broker.

Broker offre servizi di registrazione ai Server, trasmette messaggi, localizza servizi ed interagisce con altri Broker attraverso il Bridge.

Bridge incapsula funzionalità orientate alla rete e media tra il Broker locale e i Bridge dei Broker remoti.

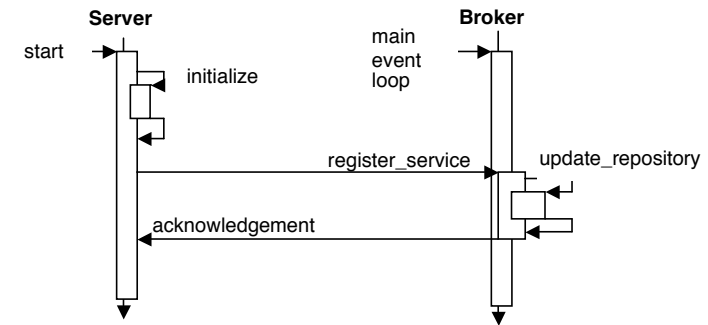
Struttura:



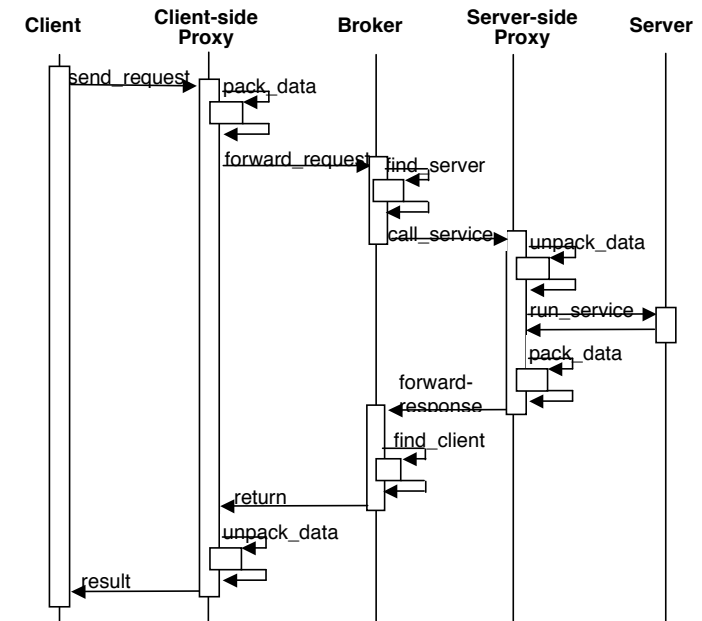
Design Pattern Broker - 3

Interazioni:

Iscrizione di un server sul Broker



Un Client fa una richiesta di servizio



Design Pattern Broker - 4

Conseguenze:

Avendo partizionato le funzionalità in componenti indipendenti, il sistema complessivo dovrebbe risultare facilmente riusabile, scalabile e modificabile.

Testare il sistema dovrebbe essere più facile avendo separato le funzionalità nei singoli componenti.

I servizi offerti sono indipendenti dalla loro posizione (location transparency).

E' supportata l'Interoperabilità tra sistemi che usano Broker.

L'indirettezza riduce l'efficienza.

Bassa fault-tolerance, se un Broker o un Server vanno giù.

Varianti:

Il Broker può avere una connessione diretta con i server.

Il Client-Proxy ottiene il riferimento al Server-Proxy e lo chiama da sé.

Le richieste possono essere trasmesse a più di un server ...

Esempi di utilizzo:

CORBA, IBM SOM/DSOM, Java Servlet, RMI, WWW