Convenzioni per la specifica

 **Nome e classificazione**

 **Sinonimi: altri nomi noti del pattern**

 Motivazione: problema progettuale

 Applicabilità: contesti in cui il pattern può essere applicato

 Struttura: rappresentazione grafica delle classi

 Partecipanti: classi e/o oggetti

partecipanti e responsabilità

 Collaborazioni tra i partecipanti

 Conseguenze: costi e benefici

 Implementazione: suggerimenti, tecniche, errori comuni

 Esempio di codice sorgente: possibile implementazione

 Utilizzo comune: il pattern nei sistemi reali

 Pattern correlati

In Breve…

|  |  |
| --- | --- |
| **NOME DEL PATTERN** | ACTIVE OBJECT |
| **CONOSCIUTO ANCHE COME** | Concurrent Object, Actor, Serializer |
| **CONTESTO** | * Applicativi che richiedono elevata concorrenza * Accesso concorrente a risorse * Necessità di creare una coda controllata * Permettere chiamate asincrone a metodi * Comunemente usato in sistemi distribuiti che richiedono server multithread |
| **PROBLEMA** | Client che devono accedere a oggetti localizzati in altri thread. |
| **SOLUZIONE** | *Active Object disaccoppia l’esecuzione di metodo dalla loro invocazione, al fine di migliorare la concorrenza e semplificare l’accesso alle risorse.* |
| **CONSEGUENZE** | **PRO**   * Disaccoppia l’esecuzione dei metodo dalla loro invocazione * Semplifica l’accesso sincronizzato a risorse condivise * *Sfrutta al Massimo il parallelismo* * *Aumento delle prestazioni*   **CONTRO**   * *Debug difficile* |
| **PATTERN CORRELATI** | * ACTOR * STRATEGY |

Problema, Contesto e Motivazioni

## INTENTO

Il Design Pattern Active Object permette di disaccopiare l’esecuzione di metodo dalla loro invocazione. Questo concetto permette a tutti gli effetti ad un client di invocare in modo asincrono un metodo, non attendendo la risposta del server. Il risultato sarà quindi disponibile in un secondo momento, una volta eseguita la risposta

## CONTESTO

In alcuni casi al fine di aumentare le prestazioni di applicazioni, scelgono di passare ad una gestione concorrente degli oggetti. Questa pratica permette di aumentare il numero di richieste dei client.

In questa configurazione l’esecuzione dei processi richiesti dal client non avviene all’interno di un thread sul client stesso. Questo pone un problema di sincronizzazione, soprattutto l’accesso ai metodi ed a risorse condivise. Per tale motivo è necessario:

* I metodi invocati non devono bloccare l’intero processo e quindi degradare le performance globale
* L’accesso agli oggetti condivisi deve essere semplice ed evitare lock o degradazione dei dati
* La richiesta dei client deve essere progettata in modo tale da sfruttare a pieno il parallelismo disponibile

## COSA OFFRE

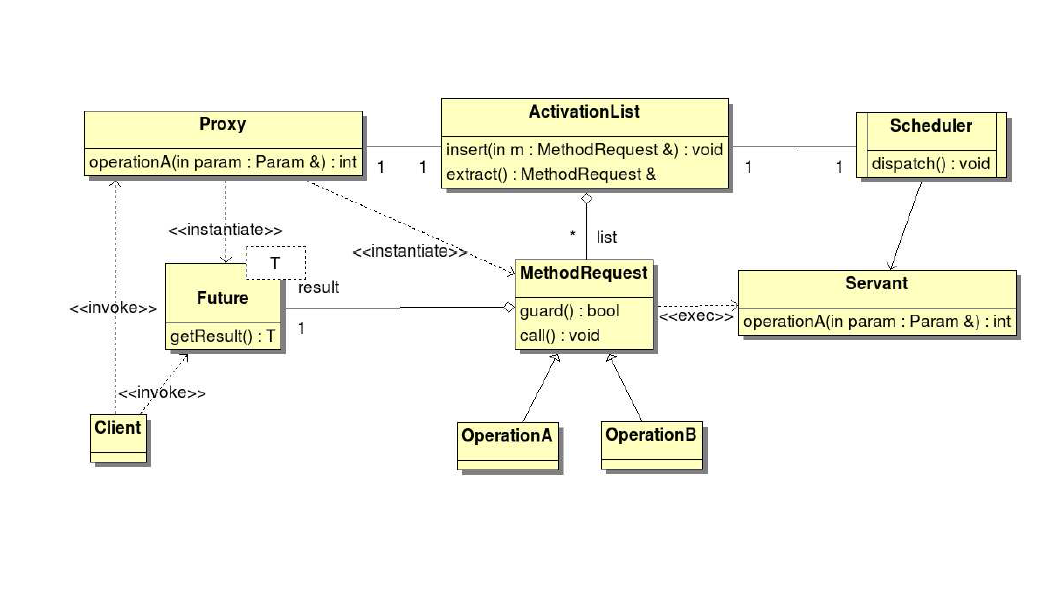
Questo design pattern offre inoltre una serie di vantaggi aggiuntivi :

* Gestione delle code
* Controllo dei parametri di richiesta
* Accesso alle risorse controllato
* Ottimizzazione del parallelismo
* Ottimizzazione della concorrenza

Active Object è ideali in tutti i contesti nei quali è necessario introdurre elevata concorrenza (code di stampa, buffer di accodamento, messaggistica, accesso ordina a risorse condivise), cercando quindi di ridurre al massimo la latenza dei client e decongestionando quindi il sistema.

STRUTTURA DEL DESIGN PATTERN

RIFARE



* **Proxy**
  + Fornisce una interfaccia al client, attraverso normali metodi
* **Method Request**
  + Una gerarchia di classi, ognuna di queste Modella una richiesta del client.
  + Deve essere presente un method request oer ogni metodo pubblico del proxy.
* **An activation list**

Contiene gli oggetti di tipo Method Request

* **A scheduler**

Gestisce la coda delle Richieste

* **A servant**

Colui che processa le richieste

* **A future**

Contiene la risposta alla richiesta del client.

#### PARTECIPANTI E LORO RESPONSABILITA’

G.

# Caso di esempio 1: Una Coda di Creazione Statistiche

Ipotizziamo di dover sviluppare un sistema di statistiche, ad esempio stampa di estratti conto per una banca. Il sistema dovrà prendere in carico le richieste dei client, interrogare il database per ottenere le informazioni, applicare delle logiche, scegliere il template corretto per la stampa, inviare per e-mail la statistica. Il processo non può essere eseguito in sincrono, in quanto il prelevamento delle informazioni (da database, web services e altre fonti dati) richiede molto tempo.

Ipotizziamo anche che il client sia un sistema indipendente dall’architettura di reportistica e non si vuol far attendere i clienti sul frontend web durante l’elaborazione.

…INSERIRE QUI IL GRAFICO…

# Caso di esempio 2: Gateway di messaggistica

Ipotizziamo di voler creare un sistema di messaggistica all’interno di una o più reti private. I client potranno spedire messaggi di testo ad una lista di utenti iscritta al servizio. Al fine di garantire maggior scalabilità con l’aumentare dei client connessi ed evitare che gli stessi client debbano aspettare che il gateway risponda o elabori la richiesta, il gateway implementerà il design pattern Active Object per aumentare il parallelismo.