Singleton

Angelo Afeltra

Intent

Si assicura che una classe abbia una sola istanza e fornisce un punto di accesso globale ad essa.

Problema

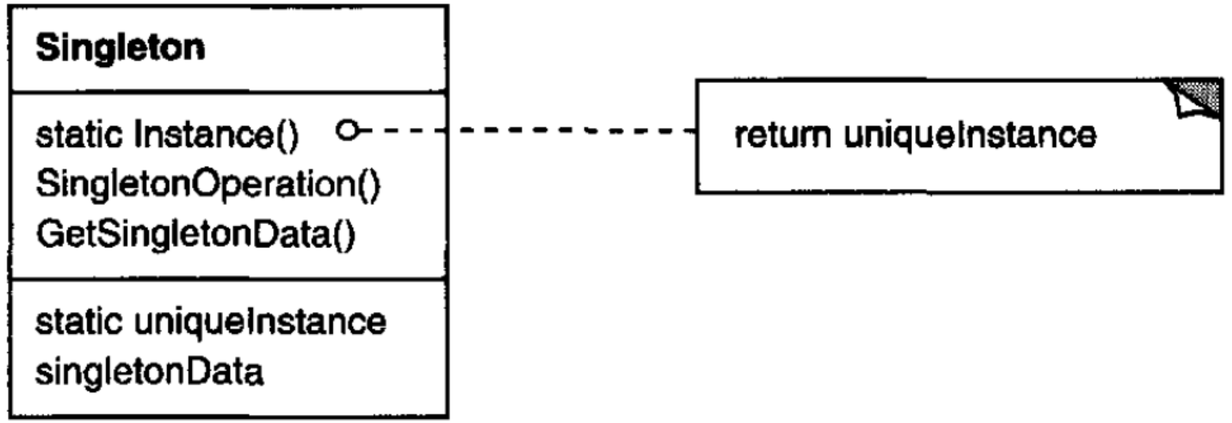
Consideriamo la necessità di controllare l'accesso a una risorsa condivisa, come un database o un file. In questa situazione, qualsiasi utente che desideri accedere a questa risorsa deve utilizzare un'istanza condivisa. L'utilizzo di un costruttore standard non è idoneo, poiché genererebbe una nuova istanza ad ogni richiesta. Tuttavia, tramite una classe singleton, è possibile sfruttare un'istanza già esistente per l'accesso alla risorsa.

Quando usare il Singleton?

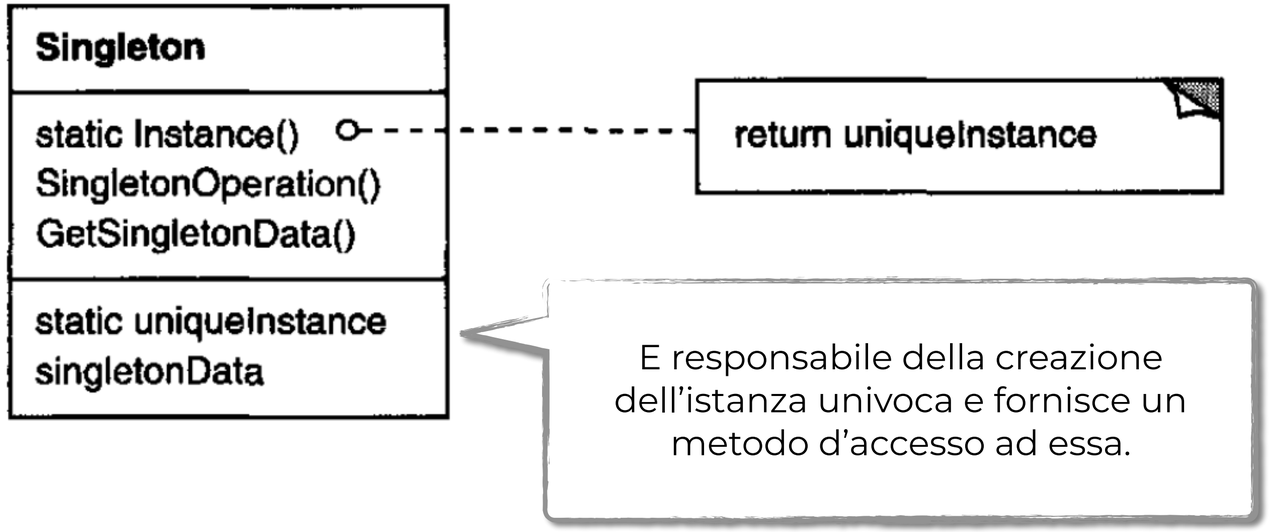
Deve esserci esattamente un'istanza di una classe e deve essere accessibile ai client da un punto di accesso ben noto.

Quando la sola istanza dovrebbe essere estendibile mediante sottoclassi e i client dovrebbero essere in grado di utilizzare un'istanza estesa senza modificare il loro codice.

Struttura


Singleton
static Instance()
SingletonOperation()
GetSingletonData()
static uniqueInstance
singletonData
return uniqueInstance

Struttura


Singleton
static Instance()
SingletonOperation()
GetSingletonData()
static uniqueInstance
singletonData
return uniqueInstance
E responsabile della creazione dell'istanza univoca e fornisce un metodo d'accesso ad essa.

Conseguenze

PRO

Accesso controllato ad una singola istanza

Si ottiene un punto di accesso globale a quell'istanza

L'oggetto singleton viene inizializzato solo quando viene richiesto per la prima volta

CONTRO

Considerato come Anti Pattern

Conseguenze

PRO

Accesso controllato ad una singola istanza

Si ottiene un punto di accesso globale a quell'istanza

L'oggetto singleton viene inizializzato solo quando viene richiesto per la prima volta

CONTRO

Poiché la classe Singleton incapsula la sua istanza unica, può avere un controllo rigoroso su come e quando i client vi accedono.

Pattern

Conseguenze

PRO

1. Accoppiamento globale: Aumenta l’accoppiamento tra i componenti
2. Ostacolo alla parallelizzazione: Richiede sincronizzazione in ambiente multithreading
3. Difficoltà nell’estensione: Rene difficile l’estendere o sostituire l’istanza
4. Creazione anticipata: L’istanza può essere creata troppo presto
5. Difficolta nel testing: Complica il testing unitario

CONTRO

Considerato come AntiPattern

Intent

Si assicura che una classe abbia una sola istanza e fornisce un punto di accesso globale ad essa.

Problema

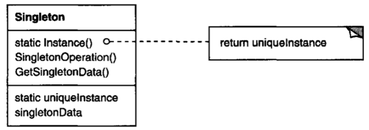
Consideriamo la necessità di controllare l'accesso a una risorsa condivisa, come un database o un file. In questa situazione, qualsiasi utente che desideri accedere a questa risorsa deve utilizzare un'istanza condivisa. L'utilizzo di un costruttore standard non è idoneo, poiché genererebbe una nuova istanza ad ogni richiesta. Tuttavia, tramite una classe singleton, è possibile sfruttare un'istanza già esistente per l'accesso alla risorsa.

Quando usare il Singleton?

Deve esserci esattamente un'istanza di una classe e deve essere accessibile ai client da un punto di accesso ben noto.

Quando la sola istanza dovrebbe essere estendibile mediante sottoclassi e i client dovrebbero essere in grado di utilizzare un'istanza estesa senza modificare il loro codice.

Struttura



Conseguenze

PRO

- Accesso controllato ad una singola istanza
- Si ottiene un punto di accesso globale a quell'istanza
- L'oggetto singleton viene inizializzato solo quando viene richiesto per la prima volta

CONTRO

- Considerato come Anti Pattern

Singleton