Design pattern: il pattern Singleton

Per definizione un pattern è “una soluzione progettuale generale a un problema ricorrente”.
In pratica, un pattern illustra il modo per risolvere un problema ricorrente che può presentarsi durante lo sviluppo di un software.

Esiste una vasta gamma di pattern, suddivisa per categorie e sottocategorie. In questo articolo tratteremo il Singleton, che appartiene alla categoria dei pattern creazionali, finalizzato ad istanziare una sola volta una determinata classe.

Presentazione

Il Singleton, pattern creazionale, ha l’obiettivo di fornire in tutto il software, una sola e unica istanza di un determinato oggetto e di fornire un punto di accesso globale ad esso.

Implementarlo è veramente semplice, bastano infatti poche righe di codice.
Ecco il diagramma UML del Singleton:

Come mostra il diagramma UML, c’è bisogno di una proprietà statica privata, un metodo statico pubblico e un costruttore privato.

In codice

Il codice per implementarlo è altrettanto breve:

La proprietà **$_instance** conterrà l’istanza della classe Singleton.
Il costruttore in questo caso è privato. Questo perché bisogna impedire di poter istanziare direttamente la classe se non tramite il metodo getInstance().
Per istanziare la classe e creare un riferimento ad essa, utilizziamo il metodo statico **getInstance()**.
La funzione di questo metodo è quella di controllare se la classe è già stata istanziata. In caso positivo, restituisce il riferimento ad essa ($_instance), altrimenti la istanzia e ne salva il riferimento.

Dalla teoria alla pratica

Creiamo la pagina index.php e scriviamo questo codice:

Abbiamo istanziato due volte e su due variabili diverse la classe Singleton con il metodo

getInstance().

Effettuando il controllo di uguaglianza sulle due variabili otterremo

bool(true)

Ciò significa che sono identiche poichè abbiamo usato l'operatore === che a differenza dell'operatore == confronta non solo se le variabili contengono lo stesso valore, ma anche che siano dello stesso tipo.

Forse ti starai chiedendo perché le due istanze sono identiche se sono effettivamente invocate due volte.
Analizziamo il codice:

Il metodo getInstance() controlla se c'è già un riferimento alla classe Singleton. Non trovandone nessuno, ne crea uno nuovo e lo salva nella proprietà $_instance.
Successivamente con questa riga:

Invochiamo nuovamente lo stesso metodo.
Se fosse stata una classe normale sarebbe stata creata una nuova istanza totalmente diversa dalla precedente, ma in questo caso, il metodo getInstance() troverà un riferimento ad una vecchia istanza della stessa classe e lo restituisce senza istanziarla nuovamente.
In poche parole, possiamo istanziare infinite volte la classe Singleton, ma lavoreremo sempre sulla stessa istanza.

Questa peculiarità ci permette di rendere l'oggetto globale. Infatti possiamo richiamare questo metodo in ogni script del nostro progetto avendo a disposizione sempre la stessa istanza di classe.

Quando non usarlo

Spesso il Singleton viene usato proprio con lo scopo di mettere a disposizione dati o variabili a livello globale, come spesso accade con il pattern Registry. Ma lo scopo per cui é nato non é esattamente questo.
Se è possibile passare per referenza una classe, non c'è nessun bisogno di renderla globale con il Singleton.

Conclusioni

In questo articolo hai conosciuto il pattern Singleton sviluppato in PHP. Ovviamente puó essere

riscritto in tutti i linguaggi.
Conoscevi già i design pattern? Come e quali utilizzi?