Taccuino:	Agile		
Creato:	29/10/2019 10.58	**Aggiornato:**	13/01/2020 16.49
Autore:	Domenico Fico		

Metadata Mapping
"*Contiene i dettagli della mappatura oggetto-relazione sotto forma di metadati*"
Visto che il codice necessario ad effettuare una mappatura oggetto-relazione può risultare essere molto ripetitivo da scrivere, il pattern Metadata Mapping permette di definirlo in un modo più semplice.
Come funziona MM?
Il problema principale è come rappresentare le informazioni all'interno dei metadati. Ci sono due principali soluzioni: generazione di codice e programmazione riflessiva.

- *Generazione di codice*:
  - scrivere un programma il cui input è il metadata e l'output è il codice sorgente della classe mappata. Questo programma responsabile della mappatura è posizionato sul server. Usando la generazione di codice bisogna assicurarsi che sia completamente integrabile all'interno del processo di costruzione senza aver bisogno di effettuare modifiche a mano.

- *Programmazione riflessiva*:
  - un programma riflessivo può chiedere un determinato metodo ad un oggetto e invocarlo passando i parametri corretti. Il programma riflessivo legge dal file metadata il nome del metodo e i parametri da passare usati per eseguire il mapping.

Un vantaggio della reflection è la sua flessibilità poiché è possibile accedere ai nomi dei campi e dei metodi di un file qualsiasi essi siano.
La reflection è abbastanza appropriata per eseguire un mapping con il db, ma è spesso sconsigliata a causa della sua lentezza. La generazione di codice è un approccio meno dinamico poiché qualsiasi cambiamento del mapping richiede la ricompilazione di almeno una parte del software (perché il mapping avviene nel processo di costruzione, prima della compilazione). Invece la programmazione riflessiva consente di apportare questo tipo di modifiche anche a rutime. Entrambi gli approcci possono risultare scomodi in fase di manutenzione.
I file metadati sono in genere scritti in file XML separati. In casi più semplici si può evitare di avere un file esterno che contiene i metadata, includendo i metadata direttamente nel codice sorgente. Un'altra alternativa è quella di contenere il mapping nel db stesso.
Quando usare MM?
Permette di ridurre il carico di lavoro necessario per gestire il db-mapping. Tuttavia è necessaria un'attenta valutazione del proprio progetto, perché in alcuni casi implementare un buon mappatore richiede molto tempo e a volte si fa prima ad eseguire il mapping da sé.

Query Object

"*Un oggetto che rappresenta una query*"

Questo oggetto consente di eseguire query senza scrivere codice SQL e senza conoscere lo schema del db sottostante.

Crea speciali metodi finder parametrizzati che nascondono codice SQL al loro interno. Questa soluzione non risulta essere adatta quando c'è bisogno di eseguire delle query specifiche.

Query Object è un *Interprete*, che è una struttura di oggetti in grado di formare query SQL. Usandolo è possibile creare query basandosi sulle classi e sui campi invece di basarsi su tabelle e colonne.

Eventuali cambiamenti dello schema del db coinvolgono solamente il Query Object.

Come funziona QO?

Il QO è un applicazione dell'*Interprete* per rappresentare query SQL. In base al sistema da implementare il QO può essere più o meno complicato. La caratteristica principale è quella di rappresentare le query attraverso il linguaggio usato (come Java) non attraverso l'SQL.

QO ha bisogno di conoscere come gli oggetti del dominio sono mappati alla struttura del db (usando il *Metadata Mapping*).

Quando usare QO?

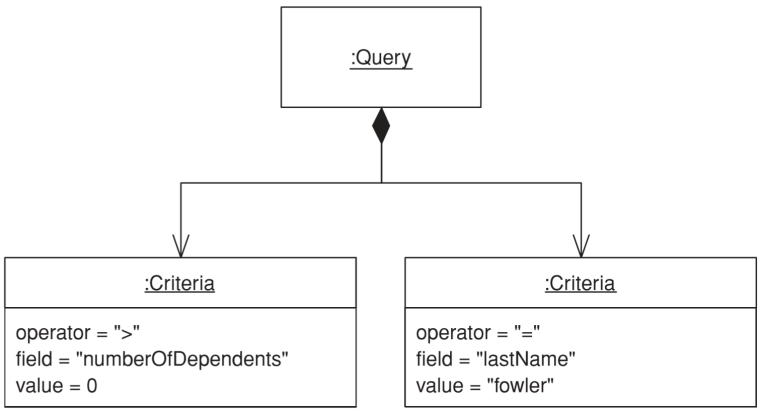
Si ha realmente bisogno del QO quando esiste un DataSource Layer, ad esempio quando si usa il *Domain Model* e il *Data Mapper*.

Per sfruttare tutte le potenzialità del QO si ha bisogno del *Metadata Mapping*.

Questi vincoli rendono il QO non adatto ad ogni progetto ed inoltre per alcuni sviluppatori è preferibile usare direttamente SQL.

I vantaggi dell'uso del QO sono visibili solamente in progetti abbastanza complessi, infatti:

- Nasconde lo schema del db,
- Supporta db multipli e schemi multipli
- Esegue ottimizzazioni evitando di eseguire più query



Repository
"Fa da mediatore tra gli oggetti del dominio e il data-mapping-layer usando un'interfaccia simile ad una collezione per accedere agli oggetti del dominio"
Un sistema con una logica di dominio complessa spesso beneficia di un livello (come quello fornito dal Data Mapper), che isola gli oggetti del dominio dal codice che accede al db. In questi sistemi spesso risulta utile avere un livello aggiuntivo responsabile della costruzione delle query, specialmente nei sistemi con molti oggetti o che hanno bisogno di query complesse.
Come funziona REP?
La Repository agisce come un mediatore tra il domain model e il livello dei mappers agendo come una collezione di oggetti. I client costruiscono le query in modo dichiarativo e le sottomettono alla Repository. Gli oggetti possono essere facilmente aggiunti e rimossi dalla Repository come se fossero in una semplice collezione, perché tutto il codice responsabile della mappatura è già all'interno della Repository.

- Fornisce una vista ancora più *object-oriented* del livello di persistenza.
- Gli oggetti del dominio non sono memorizzati all'interno della Repository
- Sostituisce i metodi finder presenti nel *Data Mapper* con un approccio basato sulle specifiche per la selezione degli oggetti
- Combina QO e MM per generare automaticamente codice SQL, per evitare che gli utenti scrivano codice SQL pensando le query direttamente in base agli oggetti.

Quando usare REP?
E' consigliato l'uso in sistemi complessi con molti oggetti nel dominio che necessitano un grande numero di query, perché riduce di molto il bisogno di scrivere una grande quantità di codice SQL.
Consente di utilizzare diverse strategie di interrogazione a seconda della situazione.