Tecnologie di Sviluppo per il Web

Programmazione su Basi di Dati: Framework Hibernate – Parte b

versione 1.0

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

- Il Concetto di Session
- Mapping e Riferimenti
- Architetture Applicative
- Aspetti Sistemistici

Il Concetto di Session

- Il concetto fondamentale di Hibernate
  - ⇨la semantica degli oggetti Session
- Programmando con JDBC
  - ⇨la programmazione è orientata all'esecuzione di singole istruoni SQL (statements)
  - ⇨il programmatore deve gestire manualmente la corrispondenza tra oggetti ed ennuple; gli oggetti diventano una vista sulla base di dati

Il Concetto di Session

- Con Hibernate
  - ⇨la filosofia cambia radicalmente
- In particolare (ATTENZIONE)
  - ⇨non è necessario pensare in termini di istruzioni SQL
  - ⇨Hibernate gestisce automaticamente la corrispondenza tra oggetti ed ennuple
  - ⇨la Session rappresenta una cache di oggetti che contengono dati sulla base di dati

Il Concetto di Session

- Session come cache
  - ⇨tutti gli oggetti su cui la Session lavora vengono messi in corrispondenza con una ennupla della base di dati
  - ⇨e vengono aggiunti ad una cache
  - ⇨la semantica della Session è quella di gestire il ciclo di vita di questa cache
  - ⇨ed in particolare la sincronizzazione con la base di dati

Il Concetto di Session

- Nota
  - ⇨la Session è una cosiddetta “cache di primo livello”
  - ⇨Hibernate ha una gestione molto più sofisticata dei meccanismi di caching
  - ⇨basati su una cache di secondo livello che è determinante per le prestazioni
  - ⇨inoltre effettua a richiesta caching degli statement

Il Concetto di Session

- Gli stati di un oggetto per Hibernate
  - ⇨"transient"
  - ⇨"persistent"
  - ⇨"detached"
- Transient
  - ⇨lo stato di un oggetto che non appartiene alla cache; il suo id è indefinito perchè non è in corrispondenza con una ennupla della base di dati

Il Concetto di Session

- Persistent
  - ⇨l'oggetto fa parte della cache e riflette lo stato di una ennupla, di cui ha il valore dell'id
  - ⇨l'oggetto può essere sincronizzato con la ennupla utilizzando update() e refresh()
- Detached
  - ⇨la cache (Session) è stata chiusa; l'oggetto mantiene il suo valore ma non è più sincronizzabile con la ennupla

Il Concetto di Session

- Il ciclo di vita dell'oggetto
  - ⇨alla creazione l'oggetto è transient
  - ⇨viene reso persistente in una sessione con save(); entra a far parte della cache
  - ⇨se viene eseguita l'operazione delete(), l'oggetto ridiventa transient, e per lavorare ancora deve essere reso di nuovo persistent
  - ⇨al termine della sessione, l'oggetto diventa detached

Il Concetto di Session

- Oggetti detached
  - ⇨Hibernate consente di "riattaccare" oggetti che precedentemente appartenevano alla cache di una Sessione chiusa alla cache di una
  - ⇨per farlo è sufficiente effettuare una qualsiasi operazione sull'oggetto nella nuova sessione
  - ⇨tipicamente: update() oppure lock()

Il Concetto di Session

- Le operazioni tipiche della Session
  - ⇨tenere traccia degli oggetti della cache
  - ⇨gestire le operazioni di sincronizzazioni periodiche con la base di dati
- In particolare
  - ⇨la Session tiene traccia delle modifiche (“automatic dirty checking”) negli oggetti della cache
  - ⇨ed effettua periodiche operazioni di “flush”

Il Concetto di Session

- flush() della Session
  - ⇨avviene periodicamente
  - ⇨ad ogni commit() di transazione
  - ⇨ad ogni chiamata del metodo flush() di Session
  - ⇨tipicamente prima dell’esecuzione delle query per garantire che il metodo list() non restituisca mai dati che non sono aggiornati

Il Concetto di Session

- Attenzione alla semantica del flush()
  - ⇨l'operazione di flush() normalmente non effettua un unico aggiornamento, ma vari aggiornamenti in modo "batch"
  - ⇨es: è in grado di aggiornare l'ennupla di un proprietario e di tutte le automobili relative
  - ⇨NON coincide con il commit(): solo dopo aver chiamato commit() le modifiche diventano permanenti

Il Concetto di Session

- Di conseguenza
  - ⇨il programmatore lavora sugli oggetti del modello
  - ⇨li modifica e – dopo averli resi persistenti – lascia che la sessione li sincronizzi con la base di dati
- Ma attenzione...
  - ⇨questo meccanismo ha delle conseguenze da tenere in considerazione

Il Concetto di Session

- Prima conseguenza: identità
  - ciascun oggetto persistente ha due identificatori (OID e id)
  - oggetti diversi (con OID diversi) possono rappresentare la stessa ennupla nella base di dati (id uguali)
  - cioè è possibile che si generino “copie” delle ennuple nella base di dati; es: creo o1, chiamo save(o1), poi eseguo una query e assegno il risultato ad o2

Il Concetto di Session

- Esistono quindi due concetti di identità
  - identità per la macchina virtuale (“object identity”: quella basata sugli OID
  - identità per la base di dati (“database identity”: quella basata sugli id
- Per Hibernate
  - conta solo la seconda
  - oggetti con OID diversi e id uguali sono considerate copie dello stesso dato

Il Concetto di Session

- Esempio
  - attenzione ad inserire due oggetti del genere in una struttura hash che non prevede duplicati (es: HashSet o HashMap)
  - bisognerebbe ridefinire equals() e hashCode() in modo da rispecchiare la database identity
  - ma non è possibile farlo sulla base dell'id (il valore dell'id cambia nel ciclo di vita)
  - e quindi bisogna utilizzare una chiave esplicita (es: targa o codice fiscale)

Il Concetto di Session

- Prima conseguenza: eccezioni
  - la semantica del caching impedisce che due copie dello stesso dato persistente siano contenute nella cache
  - Hibernate gestisce questo vincolo in modo drastico: ogni volta che una stessa sessione vede due oggetti con id uguali viene lanciata una eccezione

Il Concetto di Session

- Quindi, per fare un esempio
  - NON è possibile eseguire con Hibernate la seguente sequenza di istruzioni
  - aprire la sessione
  - rendere persistente o1 con save() (a questo punto o1 acquisisce l'id x)
  - effettuare load(o2, x) (o una query HQL)
  - a questo punto si verifica una eccezione; per evitarla bisognerebbe chiudere la sessione prima del load o della query e poi aprirne un'altra

Mapping e Riferimenti

- A questo punto
  - possiamo affrontare l'aspetto tecnico più delicato di Hibernate
  - il mapping dei riferimenti tra gli oggetti
- Il punto di partenza
  - gli oggetti con le loro associazioni
- Il punto di arrivo
  - l'implementazione nella base di dati sotto forma di chiavi esterne

Mapping e Riferimenti

- In Hibernate
  - è possibile specificare l'implementazione del mapping in modo molto fine nel file hbm
- In particolare
  - la cardinalità dei riferimenti (1 o molti)
  - il verso che si desidera mappare (monodirezionale o bidirezionale)
  - la tecnica di implementazione (con tabella intermedia)
  - la semantica del riferimento (associazione oppure aggregazione)

Mapping e Riferimenti

- Di conseguenza
  - ci sono moltissimi casi possibili, ottenute scegliendo le seguenti varianti
  - associazioni 1-1, 1-m, m-m
  - monodirezionali o bidirezionali
  - con o senza tabella intermedia (solo per 1-1 e 1-m)
  - con semantica di associazione o di aggregazione

Mapping e Riferimenti

- La sintassi del file hbm
  - ⇨elemento one-to-many
  - ⇨elemento many-to-one
  - ⇨elemento many-to-many
  - ⇨elemento one-to-one
  - ⇨elementi set, bag, list, array

Mapping e Riferimenti

- Di seguito
  - ⇨vediamo alcuni casi particolarmente interessanti
- In particolare
  - ⇨associazione 1-m senza tabella intermedia
  - ⇨associazione m-m con tabella intermedia
  - ⇨associazione 1-1 senza tabella intermedia
  - ⇨tutti i casi sono bidirezionali
  - ⇨tutti i casi utilizzano liste e bag

Mapping e Riferimenti

- Nota
  - ⇨le varianti monodirezionali si ottengono banalmente rimuovendo una parte del mapping
- Per dettagli sugli altri casi
  - ⇨il manuale di riferimento di Hibernate
  - ⇨contiene numerosissimi esempi di mapping discussi con buon livello di dettaglio

Mapping e Riferimenti

- Esempio n. 1
  - ⇨Proprietario e Automobile
  - ⇨mapping 1-m bidirezionale
  - ⇨senza l'utilizzo di tabella intermedia (viene tradotto con chiavi esterne dalla parte 1 dell'associazione)
- Il punto di partenza
  - ⇨lo sviluppo dei componenti

Mapping e Riferimenti

- In Proprietario
  - ⇨una lista di riferimenti ad automobili (con i relativi get e set, oltre ai metodi delegati)
- In Automobile
  - ⇨un riferimento al proprietario
- L'obiettivo
  - ⇨chiedere a Hibernate di gestire la corrispondenza tra i riferimenti tra i bean e le ennuple nella base di dati (salvare e caricare correttamente i riferimenti)

Mapping e Riferimenti

- Primo elemento importante
  - ⇨le associazioni bidirezionali che coinvolgono collezioni sono gestite da Hibernate in modo asimmetrico
- In particolare
  - ⇨uno dei due componenti "comanda" l'associazione ed è responsabile di salvare i riferimenti
  - ⇨l'altro funge solo da capo "inverso" per caricare i riferimenti

Mapping e Riferimenti

- Secondo elemento importante
  - ⇨Hibernate utilizza per i riferimenti una strategia di caricamento pigro
  - ⇨questo fatto pone vari vincoli sul mapping
- Il primo vincolo
  - ⇨le collezioni di associazioni devono essere dichiarate di tipo interfaccia (List, Set, Map, Collection) e non è possibile specificare implementazioni

Mapping e Riferimenti

- Infatti
  - ⇨per il caricamento pigro Hibernate utilizza proprie implementazioni interne delle interfacce di java.util
  - ⇨quando viene caricato un oggetto che ha una collezione di riferimenti, la collezione viene inizializzata con un un “proxy” vuoto di tipo interno
  - ⇨e solo successivamente i riferimenti vengono riempiti

Mapping e Riferimenti

- Nota
  - ⇨l'interfaccia corretta per la collezione di riferimenti ad automobili sarebbe Set (un proprietario non può possedere due volte la stessa automobile)
  - ⇨ma per evitare i problemi collegati ai Set, rilassiamo questo vincolo e utilizziamo java.util.List
  - ⇨il mapping viene fatto utilizzando l'elem. bag

Mapping e Riferimenti

- Il mapping di java.util.List
  - ⇨può essere fatto con l'elemento list (sequenza ordinata di riferimenti), ma in questo caso bisogna dire a hibernate di salvare in una colonna l'indice degli elementi per ripristinare l'ordine
  - ⇨oppure con l'elemento bag (multiinsieme, collezione non ordinata ma che può contenere duplicati)

Mapping e Riferimenti

>> bean + hbm + bd

- In Proprietario.hbm.xml
  - ⇨per specificare come riempire la lista, elemento bag
  - ⇨che contiene un elemento one-to-many che specifica la chiave esterna
  - ⇨NOTA: inverse="true" dice che su questa associazione comanda Automobile
- In Automobile.hbm.xml
  - ⇨per specificare come riempire il riferimento, elemento many-to-one, che corrisponde al valore di una chiave esterna

Mapping e Riferimenti

- Nota
  - ⇨per trasformare l'associazione in associazione monodirezionale è sufficiente eliminare parte del codice
- Esempio: solo da Automobile a Propriet.
  - ⇨eliminare la lista da Proprietario
  - ⇨eliminare l'elemento bag in Proprietario.hbm.xml

Mapping e Riferimenti

- Esempio n. 2
  - ⇨Autore e Libro
  - ⇨mapping m-m bidirezionale con l'utilizzo di tabella intermedia
- In questo caso
  - ⇨una lista (bag) di riferimenti in Autore
  - ⇨una lista (bag) di riferimenti in Libro
  - ⇨mapping attraverso la tabella autorilibri

Mapping e Riferimenti

>> bean + hbm + bd

- In Autore.hbm.xml
  - ⇨elemento bag
  - ⇨con elemento many-to-many
- In Libro.hbm.xml
  - ⇨elemento bag
  - ⇨con elemento many-to-many e inverse=true
- Nota
  - ⇨dal momento che i due id devono essere usati assieme nella tabella autorilibri, sono stati rinominati in idautore e idlibro

Mapping e Riferimenti

- Esempio n. 3
  - ⇨Studente e Tirocinio
  - ⇨mapping 1-1 bidirezionale senza l'utilizzo di tabella intermedia
- In questo caso
  - ⇨un riferimento in Studente al Tirocinio
  - ⇨un riferimento in Tirocinio allo Studente

Architetture Applicative

>> esempio 1-m

- Hibernate in un'architettura MVC
  - ⇨sono possibili varie soluzioni
- Soluzione n. 1
  - ⇨utilizzare la Session come DAO generalizzato
  - ⇨scrivere direttamente codice Hibernate nelle azioni
  - ⇨questo riduce al minimo lo strato di persistenza
  - ⇨svantaggio: accoppiamento delle azioni con la tecnologia di persistenza

Architetture Applicative

>> esempio m-m

- Per migliorare questa soluzione
  - ⇨è possibile arricchire il componente PersistenceManager
  - ⇨in modo che fornisca metodi per aprire e chiudere la sessione
  - ⇨e per avviare, effettuare commit e rollback delle transazioni

Architetture Applicative

- Nota
  - ⇨per la gestione della Sessione gli autori di Hibernate suggeriscono di utilizzare il pattern “ThreadLocal session”
  - ⇨mantenendo nel PersistenceManager la sessione e la transazione in variabili di tipo ThreadLocal

Architetture Applicative

- Soluzione n. 2
  - ⇨sviluppare DAO leggeri basati su Hibernate
  - ⇨lo svantaggio è che bisogna scrivere i DAO, ma il vantaggio è che le azioni sono schermate dalla tecnologia di persistenza
- Come sono fatti i DAO per Hibernate ?
  - ⇨le operazioni CRUD non hanno più senso
  - ⇨i metodi sono orientati a gestire il ciclo di vita dei bean

Architetture Applicative

>> esempio 1-1

- Tipici metodi di un DAO di Hibernate
  - ⇨metodo makePersistent()
  - ⇨metodo makeTransient()
  - ⇨metodi find vari
- In una prima soluzione
  - ⇨dal momento che la gestione delle transazioni deve essere esplicita, ciascun DAO comincia e finisce una transazione

Architetture Applicative

- E le transazioni complesse ?
  - ⇨in questa versione, per eseguire i DAO in transazioni complesse sarebbe necessario raddoppiare i metodi
  - ⇨ma in effetti non c'è nessuna necessità, dal momento che l'API è cambiata
- Una soluzione
  - ⇨gestire commit() e rollback() nelle azioni

Architetture Applicative

>> acim2pincohibernate

- I metodi del DAO
  - ⇨NON gestiscono esplicitamente le transazioni
- Le azioni
  - ⇨cominciano le transazioni
  - ⇨chiamano commit() al termine delle operazioni
  - ⇨e gestiscono l'eventuale rollback() e la chiusura della Session

Architetture Applicative

- Nota
  - ⇨nell'esecuzione dei metodi, attenzione al caricamento pigro
  - ⇨se le collezioni degli oggetti prelevati dalla base di dati non vengono esplicitamente caricate, provare ad accedere agli elementi dopo aver chiuso la sessione provoca un'eccezione (l'oggetto è detached)

Architetture Applicative

>> DAOProprietario
>> schermoStampaElenco

- Di conseguenza
  - ⇨è necessario, a richiesta, eseguire i metodi di caricamento pigro
- Per forzare il caricamento
  - ⇨metodo Hibernate.initialize(Object o)
  - ⇨eseguito su una collezione forza il caricamento dei riferimenti

Aspetti Sistemistici

>> file di build

- Tre aspetti fondamentali
  - ⇨struttura della cartella di progetto e jar necessari
  - ⇨file di build di ant
  - ⇨il target schemaexport

Riassumendo

- Il Concetto di Session
- Mapping e Riferimenti
- Architetture Applicative
- Aspetti Sistemistici

Termini della Licenza