



J2EE senza EJB, migra la tua applicazione verso Hibernate

Relatore: **Custodi Maurizio**

m.custodi@k-tech.it

(k-tech S.r.l – Javaportal)





Organizzazione del seminario

- Introduzione
- Panoramica Hibernate
- La migrazione
- Conclusioni



Introduzione

Il talk è rivolto:

a tutti gli sviluppatori, analisti, project manager ecc..

coinvolti nello sviluppo di applicazioni java
enterprise

Obiettivi del talk:

panoramica in merito alle modalità per affrontare il problema
della migrazione del motore di persistenza entity-based ad
hibernate



Panoramica su hibernate

- Hibernate standard de facto tra gli ORM open source
- Cosa fa lo strumento Hibernate?
 - Semplifica e automatizza la gestione delle entità;
 - Si occupa di rispettare le relazioni , pre-configurate, tra le entità;
 - In totale trasparenza, genera i comandi per tutte le operazioni (**CRUD**: Create, Read, Udate, Delete);
 - E' cointainer friendly, partecipa alle transazioni gestite dal cointainer;
- Semplicità di Hibernate:
 - utilizza semplici POJO (Plain Old Java Object) per il mappaggio delle entità;



La migrazione

- Domandiamoci se vale la pena:
 - Valutazione dei fattori;
- La decisione è la migrazione:
 - Hibernate in piedi in poco tempo;
 - Hibernate nell'architetturale già esistente;



La migrazione: cosa fare

- I passi da seguire per analizzare una migrazione:
 - La strategia della migrazione;
 - Disegno logico dell'applicazione;
 - Impacchettamento e configurazione;
 - Session handling;
 - EJB interaction: transaction, caching strategy, primary key generating;



*La strategia della migrazione

- Approcci possibili:
 - Big bang: migrare lo strato persistente tutto in una volta;
 - Staged: effettuare la migrazione gradualmente
 - oggetto per oggetto;
 - per funzionalità;
 - a gruppi di componenti;



Approccio di tipo Big-bang

- **Vantaggi:**
 - porting tutto in una volta;
 - uniformità del codice;
 - no preoccupazioni di collegamento entity-bean ed Hibernate;
 - semplicità di manutenzione;
- **Svantaggi:** (per i team progetto)
 - mancanza tempo;
 - mancanza risorse;
 - no tendenza a riscrivere e testare un intero strato applicativo;



Approccio di tipo Staged

- Vantaggi:
 - migrazione graduale;
- Svantaggi:
 - meno pulito in termini di disegno logico;
 - maggior complessità;
 - manutenzione di due distinte tipologie di persistenza;
 - capire quale tipologia utilizzare in ogni caso d'uso;
- Hibernate ci viene incontro, è 'Container friendly';



*Disegno logico dell'applicazione

- Modello logico che sia:
 - ben organizzato;
 - abbia responsabilità ben separate;
 - chiarezza nella collaborazione tra oggetti;
- Modello classico di architettura enterprise:
 - strato di presentazione;
 - livello di business logic;
 - livello di persistenza;
 - strato di helper;



Accelerare L'integrazione

- Due elementi chiave:
 - modello comune per rappresentare gli oggetti;
 - Centralizzazione e incapsulamento logica di persistenza;



Accelerare l'integrazione:

Modello comune di dominio

- Se non esiste, va sicuramente creato
- E' prezioso perché:
 - individua gli oggetti chiave del proprio sistema;
 - permette di progettare le tecniche di scambio tra i livelli;
 - facilita il mappaggio degli oggetti con Hibernate;
 - permette di avere sempre il controllo sull'applicazione:
 - nella manutenzione ordinaria;
 - nella manutenzione evolutiva;
 - nella gestione del cambiamento;



Accelerare l'integrazione:

Il livello di persistenza

- Tipico ambiente J2EE:
 1. entity-bean o chiamate jdbc legate allo strato business;
 - (1) cattiva progettazione, no separazione responsabilità, la migrazione avrà grande impatto.
 2. Accesso al dato incapsulato nel DAO;
 3. strato di persistenza che nasconde o DAO o entity o JDBC;
 - (2)(3) Modello migliore per la migrazione, l'implementazione è incapsulata e le responsabilità separate.
- Il livello di persistenza deve essere disaccoppiato dal livello della business logic.



* La Configurazione

- Classi principali per avvio Hibernate:
 - La Configuration:
 - crea la SessionFactory;
 - contiene le informazioni del mapping degli oggetti;
 - SessionFactory:
 - una istanza per applicazione;
 - wrappare in un singleton la creazione;



²La configurazione

- Le informazioni da fornire a Configuration:
 - all'interno del file hibernate-cfg.xml
 - (+) info di configurazione centralizzate;
 - (-) situazione poco scalare con molti oggetti, accesso simultaneo dei programmatori;
 - creare file, stesso nome, per ogni oggetto entità;
- Passiamo gli xml a Configuration:
 - programmaticamente;
 - inserendo le path nel file hibernate-cfg.xml;



*La Sessione di Hibernate

- La classe operativa fondamentale:
 - esegue tutte le operazioni sui dati;
 - apre chiude le transazioni;
- Brevi step di lavoro;
- Decidere strategia utilizzo;



*Interazione con EJB¹

- Hibernate partecipa alle transazioni CMT:
 - nulla da fare lato amministr. per far osservare la semantica e il livello di isolamento;
 - commit quando il container fa la commit;
 - idem per la roll-back;
 - Hibernate demanda l'amministrazione al container;



Caching Strategy

- Tecnica per incremento performance;
- Si può personalizzare una propria strategia;
- Due livelli di caching:
 - livello di sessione;
 - plug-in;



Concorrenza EJB e Hibernate

- Si può utilizzare;
- inconvenienti:
 - hanno meccanismi differenti;
 - le modifiche sui dati sono visibili solo dopo il flush;
 - non condividono lo spazio di memoria;
- devono usare stessa strategia di creazione PK;
 - conseguenza: duplicazione delle chiavi;
- Hibernate fornisce molte strategie per la PK;



Riferimenti utili e bibliografia

- Sito ufficiale Sun Microsystem: **www.sun.com**;
- Sito ufficiale Hibernate: **www.hibernate.org**;
- “Hibernate in Action” C. Bauer - G. King, Manning;
- Articoli javaportal: “Introduzione ad hibernate”
www.javaportal.it/docs/hibernate.htm



Ringraziamenti

- javaportal;
- k-tech S.r.l.;

Ringraziamenti speciali: Mara Mazzocchi WebMaster javaportal.it