Object-Relational Mapping

Erik Jõgi
erik.jogi@hansa.ee

Alustame lihtsa näitega

- Uue lisamine
- ID järgi laadimine
- Otsing
- Andmete muutmine

Hansabank

Objektid ja andmed

- Objektorienteeritud keeled on tunnustatult head vahendid keerulise loogikaga rakenduste loomiseks
  - Objektide abil kirjeldatakse andmekogumeid ja nende vahelisi seoseid
- Object persistance
  - objekt elab kauem kui rakendus/protsess, milles ta loodi
  - objekti "taaselustamine" teises protsessis
- Väga tihti elavad andmed tänapäeval kauem kui rakendused, milles nad loodi

Hansabank

Kuidas andmeid hoida?

- Mitterelatsioonilised lahendused
  - Object Serialization
  - Object-oriented databases
  - XML
- Relatsioonilisel andmebaasil on hulk eeliseid eelnimetatute ees
  - andmeid saab manipuleerida ilma rakendusteta
    - ei ole tarvis programmeerijaid ;-)
  - API väga paljude keelte jaoks
  - väga hea performance
  - suur hulk olemasolevaid andmeid

Hansabank

Andmed relatsioonilises andmebaasis

- Kuidas OO rakendusest neile ligi pääseda?

- madala taseme API
  - JDBC, ODBC, …
  - üldjuhul väga koodimahukas ja vaevanõudev
- kõrgema taseme SQLi vahendav API
  - **Spring**, iBATIS SQL Maps, …
  - Täiesti piisav, kui tegemist on lihtsate lahendustega

- Need lahendused ei aita meid eriti, kui me tahame kasutada korralikku objektmudelit (domain model)

Hansabank

Object-Relational Mapping - ORM

- Raamistik, mis peidab keerukuse, mis on seotud andmete salvestamisega objektidest andmebaasi ja lugemisega andmebaasist objektidesse

- Olulised osad:
  - põhiliste CRUD operatsioonide teostamise API (teek)
    - CRUD - Create, Retrieve, Update, Delete
  - keerukamate päringute sooritamise API ja/või keel (QL)
  - mapping metadata - andmete teisendamise kirjeldused

ORM raamistikud

- Kaks suurimat tegijat
  - Hibernate (JBoss) – open source
  - Toplink (Oracle) – commercial
- Standardid
  - Java Data Objects – JDO
  - Java Persistence API – JPA
    - EJB 3.0 osa

Hansabank

Moodsate ORM raamistike põhilised omadused

- Transparent persistence
  - Ei ole erilisi nõudeid objektmudelile
- Automatic dirty checking
  - Muudatused leitakse automaatselt üles (sessiooni piires)
- Transitive persistence
  - Objektide vahelisi seoseid järgitakse
- Inheritance mapping strategies
  - Erinevad võimalused objektide hierarhiate loomiseks
- Smart fetching and caching
  - Andmebaasi poole pöördutakse nii harva kui võimalik
- Development helper tools
  - Abivahendid *mapping*'u, päringute jms. tegemiseks

Hansabank

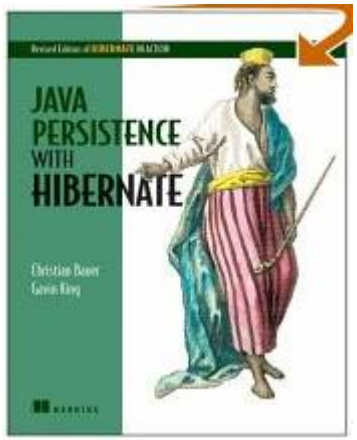
Hibernate

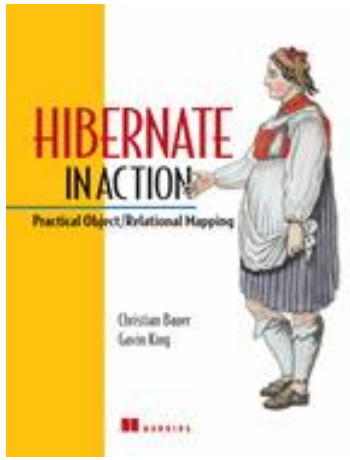
- http://www.hibernate.org

- Alustatud 2001
- Autor: Gavin King

- Täna on väljas versioon 3.2
  - core: 76 000 LoC
  - unit tests: 36 000 LoC

- Väga suur kasutajate ring
- Põhjalik dokumentatsioon
- Mitmeid raamatuid





Hansabank

Hibernate Demo

Lihtne müügisüsteem

- Klient
  - kuulub segmenti
  - ostab tooteid
- Toode
  - võib kuuluda mingisse gruppi
  - erinevatele kliendisegmentidele võib olla erinev hind

- vajadused
  - leida kõik kliendid segmendis, neid sinna lisada/ära võtta
  - leida kõik kliendi ostud kronoloogilises järjestuses
  - leida kõik grupid, kuhu toode kuulub
  - leida kõik tooted, mis gruppi kuuluvad, neid sinna lisada/ära võtta
  - leida toote hinnad erinevate segmentide lõikes
  - vajadus leida kõik ostud kuupäeva järgi

Hansabank

Objektmudel

- Customer
  - int ID
  - String name
  - Segment
  - SortedSet<Purchase>
- Segment
  - int ID
  - name
  - Set<Customer>
- Group
  - int ID
  - String name
  - Set<Product>
- Product
  - int ID
  - String name
  - BigDecimal price
  - Set<Group>
  - Map<Segment, BigDecimal>
- Purchase
  - int ID
  - Date date
  - Customer
  - List<Purchase.Row>
- Purchase.Row
  - Product
  - int quantity
  - BigDecimal price

Hansabank

Andmemudel

- CUSTOMER
  - **ID**
  - NAME
  - SEGMENT_ID
- SEGMENT
  - **ID**
  - NAME
- GROUP
  - **ID**
  - NAME
- GROUP_PRODUCT
  - **GROUP_ID**
  - **PRODUCT_ID**
- PRODUCT
  - **ID**
  - NAME
  - PRICE
- PRODUCT_PRICE
  - **PRODUCT_ID**
  - **SEGMENT_ID**
  - PRICE
- PURCHASE
  - **ID**
  - DATE
  - CUSTOMER_ID
- PURCHASE_ROW
  - **PURCHASE_ID**
  - **_INDEX**
  - **PRODUCT_ID**
  - QUANTITY
  - PRICE

Hansabank

Collection mapping

- Segment.customers
  - Set
- Customer.purchases
  - SortedSet
  - sorteerimine erinevate kriteeriumide alusel
- Group.products
  - Set
  - eraldi tabel
- Product.prices
  - Map
  - eraldi tabel
- Purchase.rows
  - List
  - Row ei ole entity

Hansabank

Bi-directional mapping

- many-to-one
  - Customer.segment
  - Segment.customers

  - Purchase.customer
  - Customer.purchases

- many-to-many
  - Product.groups
  - Group.products

Hansabank

Lazy loading

- The 1+N problem
  - find all customers -> Customer.purchases
- Proxies vs. fetching
  - find all customers -> Customer.segment
- Proxy warning:
  - do not access variables directly!

Hansabank

Caching

- Session cache
  - find purchase
  - find another purchase
  - if the customer is the same then it is only loaded once
  - queries always go to the database
- Second-level cache
  - different implementations
  - different strategies
  - query cache

Hansabank

Queries

- Criteria API
  - find all purchases made by customer X or Y before day Z
  - better support for refactoring
- HQL – Hibernate Query Language
  - find all customers who bought product X
  - allows very complex queries

Hansabank

Java access modifiers

- public, proteced, <package private>, private
  - Hibernate doesn't care
    - but CGLIB cares sometimes!
- bi-directional mappings
  - separate methods that modify both ends
- property vs. field access

Hansabank

JPA – Java Persistance API

- Sun'i poolt kehtestatud standard
- Välja töötamisel osalesid aktiivselt Hibernate'i ja Toplink'i arendajad
- Tulemus on väga sarnane Hibernate'ga
- Hibernate'iga tuleb kaasa ka JPA interface'de implementatsioon

- Ainult alates Java SE 5.0
- Kasutab Java annotations
  - http://www.hibernate.org/hib_docs/annotations/reference/en/html_single/

Hansabank