Portál pre odborné publikovanie ISSN 1338-0087

Hibernate

Hladik Ivan · Informačné technológie

11.03.2011

Hibernate 3.0 je open-source nástroj (LGPL licencia) pre objektovo-relačné mapovanie tried v jazyku Java na tabuľky relačnej databázy. Na popis rozloženia objektov do tabuliek sa využívajú XML súbory, alebo v tejto dobre preferovanejšie anotácie (Hibernate Annotations). Následne pomocou jazyka HQL alebo Hibernate Criteria Query API je možné data z databáze vyberať v podobe objektov čo významne redukuje čas potrebný pre vývoj aplikácie.

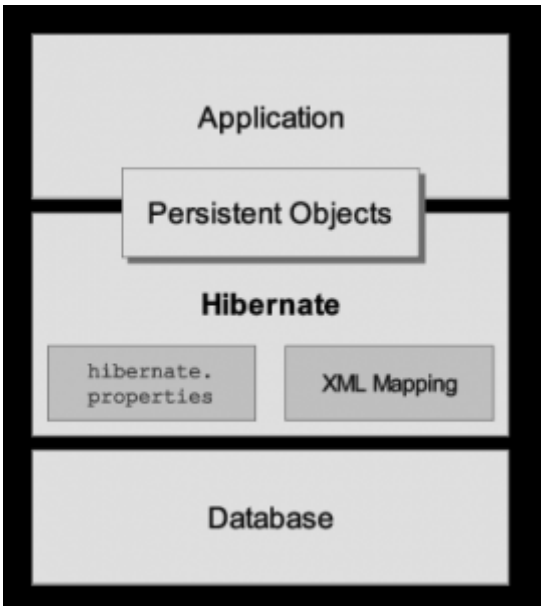
Dovoľuje transparentnú perzistenciu, čo umožňuje aplikáciám zmenu na iné databázy. Poskytuje dobre navrhnuté mechanizmy pre tranzakčné spracovávanie a aj caching dotazov.Hibernate je možné použiť pri Java Spring, Java Servlet-based, J2EE aplikáciách , alebo pri aplikáciách využívajúcich EJB „session beens“. Ďalšie znaky :

- Criteria Query API s podporou projekcie, agregovanie a subselektov.
- Monitorovanie výkonnosti cez JMX alebo lokálneho Java API
- Je škálovateľný a vhodný do clustrových prostredí.
- Pre vývojárov je pripravená kompletná podpora do Eclipse.
- Redukuje čas na vývoj využitím dedičnosti, polymorfizmu a Java Collection frameworku.
- Podporuje automatické generovanie primárnych kľúčov.
- Objektovo-relačné mapovanie:
  - 3 rozdielne stratégie O/R mapovania
  - Polymorfické asociácie
  - Obojsmerné asociácie
  - Filtrovanie
  - Indexované kolekcie
- Hibernate Dual-Layer Cache akchitektúra (HDLCA)
  - Zabezpečenia vlákien
  - Neblokuje prístup k dátam
  - Session level cache
  - Second-level cache
  - Query cache
- Výkonnosť
  - Lazy inicializácia
  - Dávkové načítavanie
  - Podpora optimistických zámkov s verzionovaním a časovými známkami
  - Vysoko škálovateľná architektúra

- SQL generované v čase inicializácie systému
- Connection pooling
- Prepared statement caching

Architektúra

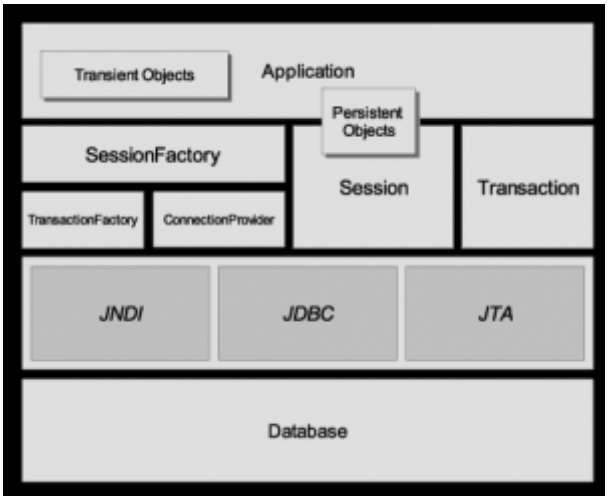
Aplikácia využívajúca Hibernate je tvorená tromi vrstvami a to dátovou, aplikačnou a prezentačnou. Dátová vrstva, resp. DAO (Data Access Objects) vrstva je tvorená sadou tried, ktoré sa starajú o manipuláciu perzistentných dát nad databázou. Je jediná, ktorá komunikuje priamo s Hibernate.



Hibernate

Z diagramu vidieť, že Hibernate využíva databázu a konfiguračné súbory na to, aby pre aplikáciu zabezpečil perzistenciu servisov a perzistentné objekty. Hibernate je konfigurovateľný pomocou nastavení v „hibernate.properties“ a pomocou XML súboru reprezentujúceho mapovanie POJO (Plain Old Java Object) objektov na databázovú schému. XML súbor môže taktiež definovať perzistentné vlastnosti a asociácie objektov. Hibernate komunikuje s databázou priamo cez JDBC (Java Database Connectivity) spojenia. Hibernate architektúra sa skladá z troch základných komponentov:

- Connection Mangement – Otváranie a zatváranie spojení s databázou si vyžaduje veľké množstvo vynaložených zdrojov, je to najdlhšie trvajúca činnosť interakcie s databázou, preto Connection Management má za úlohu zabezpečiť efektívne riadenie spojení s databázou.
- Transaction Management – Umožňuje používateľovi vykonávať viac než jeden dopyt na databázu v rovnakom čase.
- Object Relation Mapping – Objektovo-relačné mapovanie je technika mapovania kde sú dáta reprezenetované z objektového modelu do modelu relačného. Táto časť Hibernate sa využíva na select, insert, update a delete záznamov z tabuliek. Keď sa posunie objekt metóde session.save(), Hibernate načíta stav premenných objektu a vykoná potrebné query.



Hibernate architecture detail

SessionFactory (org.hibernate.SessionFactory)

SessionFactory je mapovaná pomocou nastavení v hibernate.properties. Obsahuje XML mapovanie tried, predlohu pre session a poskytovateľa spojenia. Ak aplikácia pracuje s viacerými databázami, pre každú musí byť vytvorená separátna SessionFactory.

Session (org.hibernate.Session)

Je to jednovláknový objekt krátkej životnosti reprezentujúci konverzáciu medzi aplikáciou a perzistentným úložiskom dát. Obaľuje JDBC spojenie a slúžia ako factory pre tranzakcie, pričom udržuje cache prvej úrovne.

Persistent objects and collections

Sú jednovláknové objekty obsahujúce perzistentný stav a business funkciu. Väčšinou sú to JavaBean-y alebo POJO. Sú asociované práve na jednu session. Keď sa session uzavrie, objekt sa odpojí a uvoľní pre ďalšie využitie v aplikačnej vrstve. (Npr. priamo ako DTO medzi prezentačnou a aplikačnou vrstvou.)

Transient and detached objects and collections

Sú inštancie perzistentných tried, ktoré práve nie sú asociované so session.

Transaction (org.hibernate.Transaction)

Rozhranie Transaction je doporučené API, ale nie je povinné .Tranzakcia je jednovláknový objekt krátkej životnosti využívaný aplikáciou na špecifikovanie základných jednotiek na prácu. Slúži na abstrakciu aplikácie od JDBC, JTA alebo COBRA tranzakcie.

ConnectionProvider (org.hibernate.connection.ConnectionProvider)

(Voliteľné) Je factory pre JDBC spojenia. Abstrahuje aplikáciu od nižšie ležiacich Datasource a DriverManager-a. ConnectionProvider nie je vystavený aplikácii ale môže byť rozšírený a implementovaný vývojárom.

TransactionFactory (org.hibernate.TransactionFactory)

Je factory pre inštancie Transaction. Nie je vystavená aplikácii, ale môže byť rozšírená a implementovaná vývojárom.

Extension Interfaces

Hibernate ponúka sadu možných rozšírení pre úpravu implementácie perzistentnej vrstvy.

Konfigurácia Hibernate

Konfiguračný súbor

Nastavenie konfiguračného súboru hibernate.cfg.xml, ktorý je potrebné umiestniť do root directory CLASSPATH, kde ho Hibernate hľadá pri spustení.

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
<session-factory>
<property name="connection.username">user</property>
<property name="connection.password">pwd</property>
<property name="connection.url">jdbc:mysql://localhost/dbName</property>
<property name="connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="dialect">org.hibernate.dialect.MySQL5InnoDBDialect</property>
<property name="hibernate.connection.pool_size">10</property>
<property name="hibernate.show_sql">true</property>
<property name="hibernate.hbm2ddl.auto">update</property>
</session-factory>
</hibernate-configuration>
```

Dialect :

- DB2 – org.hibernate.dialect.DB2Dialect
- HypersonicSQL – org.hibernate.dialect.HSQLDialect
- Informix – org.hibernate.dialect.InformixDialect
- Ingres – org.hibernate.dialect.IngresDialect
- Interbase – org.hibernate.dialect.InterbaseDialect
- Pointbase – org.hibernate.dialect.PointbaseDialect
- PostgreSQL – org.hibernate.dialect.PostgreSQLDialect
- Mckoi SQL – org.hibernate.dialect.MckoiDialect
- Microsoft SQL Server – org.hibernate.dialect.SQLServerDialect
- MySQL – org.hibernate.dialect.MySQLDialect
- Oracle (any version) – org.hibernate.dialect.OracleDialect
- Oracle 9 – org.hibernate.dialect.Oracle9Dialect
- Progress – org.hibernate.dialect.ProgressDialect

- FrontBase – org.hibernate.dialect.FrontbaseDialect
- SAP DB – org.hibernate.dialect.SAPDBDialect
- Sybase – org.hibernate.dialect.SybaseDialect
- Sybase Anywhere – org.hibernate.dialect.SybaseAnywhereDialect

Perzistentné triedy

Perzistentné triedy sú triedy aplikácie, ktoré reprezentujú entity modelované pre danú aplikačnú oblasť. Je vhodné, aby tieto triedy boli vytvorené ako POJO. Príklad perzistentnej triedy (mapovanie pomocou anotácií, vyžaduje sa package HIbernate Annotations):

```
@Entity
@Table(name="usertbl")
public class User implements Serializable{

public Transaction() {}

@Id
@GeneratedValue
@Column(name="id")
private Integer id;

@Column(name="name")
private String name;

public Integer getId() {
return id;
public void setId(Integer id) {
this.id = id;

public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
}
}
```

Práca s objektami v aplikácii a HelloHibernate.java

Automatické načítanie vlastností zo súboru hibernate.cfg.xml

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
```

```
public class HellHIbernate {
public static void main(String[] args) {

Session session = null;
Transaction tx = null;

try{

Configuration cfg = new Configuration();
SessionFactory factory = cfg.configure().buildSessionFactory();
session =sessionFactory.openSession();
tx = session.beginTransaction();

System.out.println(“Create User”);
User user = new User();
user.setId(1);
contact.setName(“First User”);
session.save(user);
Tx.commit();
System.out.println(“Done”);

}catch(Exception e){
if(tx != null) tx.rollback;
}finally{
session.flush();
session.close();
}

}
```

Dotazy na perzistentné objekty

Jazyk HQL

Je plne objektovo orientovaným dotazovacím jazykom. Jeho syntax sa veľmi podobá na SQL a podobne ako SQL je aj case-insensitive, okrem mien tried a vlastností. HQL používa triedy a vlastnosti namiesto tabulie a stĺpcov. Je veľmi silný, umožňuje využívať polymorfizmus a asociácie. Výhody:

- Plná podpora pre relačné operácie: HQL umožňuje reprezentovať SQL dotazy vo forme objektov.
- Vracia výsledok ako objekt: HQL query vrátia výsledok dotazu vo forme objektu, ktorý je ľahko ovládateľný.
- Polymorfné dotazov: HQL plne podporuje polymorfné query. Výsledky polymorfného dotazu sú objekty spolu so všetkými podriadenými objektami, ak existujú.
- Podpora funkcií: HQL obsahuje mnoho funkcií, ako je stránkovanie, fetch join s dynamickým profilovaním, inner/outer/full join, karteziánske produkty. Tiež podporuje projekciu , agregáciu (max, avg) , zoskupovanie a triedenie.
- Databázovo nezávislé: dotazy napísané v HQL sú nezávislé na databáze (Ak databáza podporuje základné funkcie).

Príklad:

List users = null;
users = session.createQuery(“from User”).list();

Criteria

Vyhľadávacie kritériá poskytujú alternatívny spôsob pre získavanie perzistentných objektov. Dotaz sa zostavuje dynamickým pridávaním podmienok.

Príklad:

criteria.add(Restriction.between(“id”, new Integer(100),
new Integer(200));
criteria.addOrder(Order.asc(“name”));
List users = criteria.list();

Zdroje:

1. http://www.hibernate.org/
2. http://docs.jboss.org/
3. Hibernate Tutorials, Gary Mak 2006