KIV/DB2 - Objektově relační mapování: Hibernate - data

Manipulace s daty

Pro úspěšné zvládnutí tohoto cvičení je nutné mít vytvořený projekt ze cvičení minulého. V tomto projektu byl navržen model dvou tříd `Koktejl` a `Prisada`, které byly spojeny asociací typu M:N. Obě třídy obsahovaly základní atributy název a cena a pro jednoznačnou identifikaci také celočíselný atribut `Id`. K těmto třídám byly automaticky vytvořeny odpovídající mapovací soubory a po korektní konfiguraci frameworku Hibernate byly ve vybraném SŘBD vytvořeny odpovídající tabulky. Pro práci použijte svůj projekt z minulého cvičení nebo si stáhněte HiberObjects projekt *Bar* včetně odpovídajícího JDBC driveru do vývojového prostředí Eclipse s instalovaným pluginem HiberObjects. Stažený archiv rozbalte a celou strukturu importujte jako existující projekt Eclipse.

Pozn. Pokud importovaný projekt vykazuje chyby, je nutné do projektu zahrnout všechny požadované knihovny. Toho docílíme ve vlastnostech projektu, záložka HiberObjects/persistence kde "občerstvíme" políčka Add Hibernate libraries a Add HSQLDB libraries.

Připravený HiberObjects projekt *Bar* doplníme o novou třídu `Bar`, která definuje metody pro práci s perzistentními objekty uloženými v databázi. Staženou třídu importujte do programového balíku `db2`. Spusťte metodu `main` třídy `Bar` a prozkoumejte obsah databáze.

Data Access Objects vrstva

Pokud model tříd projektu neobsahuje pro třídy `Koktejl` a `Prisada` prislusne *DAO* třídy, vytvořte je. Právě metodami této třídy je správné přistupovat k perzistentním objektům uložených v databázi. Každá DAO třída poskytuje tyto **CRUD** operace:

create
: metoda, která uloží příslušný objekt do databáze. V příslušné tabulce se vytvoří jeden záznam. Pokud ukládaný objekt obsahuje množinu jiných objektů, jsou tyto objekty také uloženy nebo aktualizovány.

read
: metoda, která načte příslušný objekt z databáze podle jeho `Id`. Může se jednat buď o jeden záznam z jedné tabulky nebo navíc odpovídající záznamy z podřízených tabulek.

update
: metoda, která aktualizuje příslušný objekt v databázi. Převážně se jedná o nastavení/změnu/zrušení hodnoty cizího klíče nebo vložení/smazání záznamu do/v rozkladové tabulce.

delete
: metoda, která smaže příslušný objekt v databázi. To znamená, že je smazán pouze jeden záznam z příslušné tabulky. Pokud existuje restriktivní referenční omezení, dojde během operace k chybě.

Získání otevřeného spojení do databáze

Každá manipulace s perzistentními objekty je realizovatelná pouze nad otevřeným spojením (tzv. *session*) do databáze. Získání této session je definováno ve třídě `HibernateHelper` v balíku `util` metodou `getInstance()`. Spojení s databází získáme příkazem:

```
Session s = HibernateHelper.getInstance().getSession();
```

Transakční zpracování

Pokud chceme např. vytvořený objekt uložit do databáze jako perzistentní, musíme volat metodu `create` v transakci. Začít novou transakci lze pouze nad otevřených spojením s databází. Transakci vytvoříme příkazem:

```
s.beginTransaction();
```

V těle transakce se předpokládá volání CRUD metod nad objekty a ukončení transakce (její spáchání) je voláno metodou `commit()`.

```
s.getTransaction().commit();
```

Získání perzistentních objektů z databáze

Také získání požadovaného objektu z databáze vyžaduje aktivní spojení s databází. Framework Hibernate nabízí programátorovi několik možností, jak tento objekt získat. Mezi základní patří:

- využití interního dotazovacího jazyka **HQL**
- definování tzv. *kriterií* pomocí Java rozhraní `Criteria`, `Criterion` a třídy `Restrictions`, ...

Získání dat - jazyk HQL

Dotaz, kterým chceme získat požadovaná data (objekty) je formulován jazykem HQL, který ve své syntaxi vychází z jazyka SQL. V dotazu se neodkazujeme na tabulky, ale na názvy tříd a jejich atributů. Následující kód ukazuje získání všech koktejlů:

```
Session s = HibernateHelper.getInstance().getSession();
Query query = s.createQuery("from Koktejl");
List<Koktejl> koktejly = query.list();
```

Pokud budeme chtít získat všechny přísady vybraného koktejlu podle jeho `koktejl_id`, sestavíme požadovaný dotaz takto:

```
Session s = HibernateHelper.getInstance().getSession();
Query query = s.createQuery("from Prisada as p inner join fetch p.koktejl as k where k.id = :koktejl_id");
query.setLong("koktejl_id", koktejl_id);
List<Prisada> prisady = query.list();
```

Získání dat - kritéria

Dotaz, kterým chceme získat požadovaná data (objekty) je formulován jednoduchým kritériem, chceme získat všechny koktejly:

```
Session s = HibernateHelper.getInstance().getSession();
Criteria crit = s.createCriteria(Koktejl.class);
List<Koktejl> koktejly = crit.list();
```

Pokud budeme chtít získat všechny přísady vybraného koktejlu podle jeho `koktejl_id`, sestavíme požadovaný dotaz takto:

```
Session s = HibernateHelper.getInstance().getSession();
Criteria crit = s.createCriteria(Prisada.class);
crit.createCriteria("koktejl", "k");
crit.add(Restrictions.eq("k.id", koktejl_id));
List<Prisada> prisady = crit.list();
```

Příkazem `crit.createCriteria("koktejl", "k");` do dotazu zahrnujeme všechny koktejly, kde se používá daná přísada. Vybrání koktejlu s požadovaným `koktejl_id` zajistí podmínka selekce příkazem `crit.add(Restrictions.eq("k.id", koktejl_id));`

Dokončení konfigurace frameworku Hibernate

V minulém cvičení byla ukázána základní konfigurace frameworku v souboru .properties. Nyní konfiguraci dokončíme nastavením nových parametrů. V prvé řadě zakomentujeme parametr `hibernate.hbm2ddl.auto`, jinak bude stále při volání funkce main třídy `Bar` databáze včetně dat zničena a znova založena. Jakmile budou v programu odladěny dotazy, je též vhodné zakázat zobrazování generovaného SQL kódu.

```
#hibernate.hbm2ddl.auto=create-drop

hibernate.show_sql=false
```