Hibernate is equipped with an extremely powerful query language that (quite intentionally) looks very much like SQL. HQL is fully object-oriented, understanding notions like inheritance, polymorphism and association.

Queries are case-insensitive, except for names of Java classes and properties. The following some of the queries related to Publisher and Title classes.

The following are the entities we use in the query examples:

**Publisher.java**

```java
package rel;
import java.util.Set;
public class Publisher {
    private int pid;
    private String pname;
    private Set titles;
    public Publisher() {}
    public Publisher(String pname) {
        this.setPname(pname);
    }
    public int getPid() {
        return pid;
    }
    public void setPid(int pid) {
        this.pid = pid;
    }
    public String getPname() {
        return pname;
    }
    public void setPname(String pname) {
        this.pname = pname;
    }
    public String toString() {
        return pid + "::" + pname;
    }
    public Set getTitles() {
        return titles;
    }
    public void setTitles(Set titles) {
        this.titles = titles;
    }
}
```

**Title.java**

```java
package rel;
public class Title {
    private int titleid;
    private String title;
    private String author;
    private double price;
    private Publisher publisher;
    public Title() {
    }
    public Title(String title, String author, double price, Publisher publisher) {
        this.title = title;
        this.author = author;
        this.price = price;
        this.publisher = publisher;
    }
    public int getTitleid() {
        return titleid;
    }
}
```
public void setTitleid(int titleid) {
    this.titleid = titleid;
}

public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

public String getAuthor() {
    return author;
}

public void setAuthor(String author) {
    this.author = author;
}

public double getPrice() {
    return price;
}

public void setPrice(double price) {
    this.price = price;
}

public Publisher getPublisher() {
    return publisher;
}

public void setPublisher(Publisher publisher) {
    this.publisher = publisher;
}

public String toString() {
    return titleid + "": " + title + ":" + author + ":" + price + ":"
            + publisher.getPname();
}

DiscountTitle.java

package rel;
public class DiscountTitle extends Title {
    private double disrate;

    public DiscountTitle() {
    }

    public DiscountTitle(String title, String author, double price, Publisher publisher, double disrate) {
        super(title, author, price, publisher);
        this.setDisrate(disrate);
    }

    public double getPrice() {
        return super.getPrice() - super.getPrice() * getDisrate();
    }

    public double getDisrate() {
        return disrate;
    }

    public void setDisrate(double disrate) {
        this.disrate = disrate;
    }

    public String toString() {
        return super.toString() + ":" + disrate;
    }
}
<xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="rel.Title" table="TITLES" discriminator-value="Title">
    <id name="titleid" column="TITLEID">
      <generator class="native"/>
    </id>
    <discriminator column="booktype" type="string" />  
    <property name="title" type="string" column="TITLE"/>
    <property name="author" type="string" column="AUTHOR"/>
    <property name="price" type="double" column="PRICE"/>
    <many-to-one
      name="publisher"
      column="pid"
      class="rel.Publisher" />  
  </class>

  <class name="rel.Publisher" table="PUBLISHERS">
    <id name="pid" column="PID">
      <generator class="native"/>
    </id>
    <property name="pname" type="string" column="PNAME"/>
    <set name="titles" inverse="true">
      <key column="PID"/>
      <one-to-many class="rel.Title"/>
    </set>
  </class>

  <subclass name="rel.DiscountTitle" extends="rel.Title" discriminator-value="Discounted">
    <property name="disrate" type="double"/>
  </subclass>
</hibernate-mapping>

Insert data into tables – PUBLISHERS and TITLES, using the following commands.

```sql
delete from titles;
delete from publishers;
insert into publishers values (1,'Apress');
insert into publishers values (2,'Willy');
insert into publishers values (3,'Wrox');
insert into publishers values (4,'Manning');
insert into titles values(1,'Pro Spring','Rob Harrop',450,1,'Title',null);
insert into titles values(2,'Mastering EJB 3.0','Rima Patel',550,2,'Title',null);
insert into titles values(3,'Pro EJB 3','Mike Keith',399,1,'Title',null);
insert into titles values(4,'Beginning Hibernate','Dave Minter',550,3,'Discounted',.40);
insert into titles values(5,'Hibernate in Action','Gavin King',500,3,'Title',null);
insert into titles values(6,'Spring in Action','Craig Walls',450,3,'Discounted',.20);
```
public class HQLServlet extends HttpServlet {
    Session session = null;
    public void init() {
        session = new Configuration().configure().buildSessionFactory().openSession();
    }
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            String query = request.getParameter("query");
            if (query == null) query = "";
            out.println("<form action='hql' method='post'>Enter query below : <br/>
            <textarea name='query' rows='5' cols='50'>" + query + "</textarea>"");
            if (query.length() > 0) {
                List lst = session.createQuery(query).list();
                for (Object obj : lst) {
                    if (obj instanceof Object[]) {
                        for (Object o : (Object[]) obj)
                            out.println(o);
                    } else
                        out.println(obj.toString() + "<br>");
                }
            } finally {
                out.close();
            }
        }
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }
}
Expressions

- Mathematical operators: +, -, *, /
- Binary comparison operators: =, >=, <=, <>, !=, like
- Logical operations: and, or, not
- Parentheses: ( ), indicating grouping
- in, not in, between, is null, is not null, is empty, is not empty, member of and not member of
- "Simple" case, case ... when ... then ... else ... end, and "searched" case, case when ... then ... else ... end
- String concatenation: ...||... or concat(...,...)
- Mathematical functions: current_date(), current_time(), current_timestamp()
- Mathematical functions: second(...), minute(...), hour(...), day(...), month(...), year(...), locate(), abs(), sqrt(), bit_length(), mod()
- Logical operations: and, or, not
- Parentheses: ( ), indicating grouping
- String functions: coalesce() and nullif()
- Mathematical functions: str() for converting numeric or temporal values to a readable string
- Mathematical functions: cast(... as ...), where the second argument is the name of a Hibernate type, and extract(... from ...) if ANSI cast() and extract() is supported by the underlying database
- HQL functions that take collection-valued path expressions: size(), minelement(), maxelement(), minindex(), maxindex(), along with the special elements() and indices functions which may be quantified using some, all, exists, any, in.
- JDBC-style positional parameters: ?
- JDBC-style named parameters: :name, :start_date, :x1
- SQL literals: 'foo', 69, 6.66E+2, '1970-01-01 10:00:01.0'
- Java public static final constants: eg. Color.TABBY

Examples

The following are example of HQL using Publisher, Title and DiscountTitle entities.

```hql
from Title
from Title as t
from Title where price > 500
from Title where title like 'Pro%'
from Title where instr(title,'EJB') > 0
select t.title || '-' || t.author from Title t
select max(price) from Title
select t.publisher.pname, max(t.price) from Title t group by t.publisher.pname
from Title where publisher.pid = 1
from Title t where t.disrate is not null
from Title t where t.publisher.id in (1,2)
from Title t where t.price = ( select max(t.price) from Title t)
select t.publisher from Publisher where size(titles) > 1
select t.publisher from Title t where booktype = 'Discounted'
from Publisher p where not exists ( from Title where publisher.pid = p.pid)
select floor(price * 0.15) from Title
```
Native SQL

You may also express queries in the native SQL dialect of your database. Hibernate3 allows you to specify handwritten SQL (including stored procedures) for all create, update, delete, and load operations.

```java
sess.createSQLQuery("SELECT * FROM TITLES").list();
sess.createSQLQuery("SELECT TITLE, PRICE FROM TITLES").list();
```

Named Query

It allows you to store a query in mapping file and refer to it from your application. It enables you to modify the query without changing source code of your application.

```xml
<hibernate-mapping>
  <class name="Account" table="ACCOUNT">
    . . .
  </class>
  <query name="Account.LowBalanceAccounts">
    <![CDATA[ select a from Account a where a.balance < 5000 ]]> 
  </query>
</hibernate-mapping>
```

The following code gets a names query with the given name and then executes it.

```java
result = session.getNamedQuery("Account.LowBalanceAccounts").list();
for( Object obj : result) {
  // process
}
```

Query by Example (QBE)

QBE allow you to obtain the result by providing sample data regarding the data you want. You populate an instance with the required data and then hibernate creates query with the data provided by you in the instance.

**Example** class, which implements **Criteria** interface, contains the QBE functionality.

```java
Criteria c = session.createCriteria(Account.class);
Account sample = new Account();
sample setCname("Srikanth");

Example example = Example.create(sample);
example.excludeProperty("acno");
example.excludeProperty("balance");
c.add(example);

List result =c.list();
for( Object obj : result) {
  // process
}